Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

More Applicable Text Classification with Human-in-the-Loop: Patterns, Frameworks, and Tools

Dissertation with the aim of achieving a doctoral degree **(Dr. rer. nat.)**
at the Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Applied Software Technology
Universität Hamburg

submitted by

Jakob Smedegaard Andersen

from Hamburg

Hamburg, 2024

Day of the oral defense:	July 19, 2024
Chair of the examination board:	Prof. Dr. Sören Laue Prof. Dr. Matthias Rarey (deputy)
First reviewer:	Prof. Dr. Walid Maalej
Second reviewer:	Prof. Dr. Olaf Zukunft
Third reviewer:	Prof. Dr. Foutse Khomh

To My Family.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. Walid Maalej, for his tireless support, guidance, and expertise throughout my Ph.D. Thank you for giving me the opportunity to be a part of your research group and for constantly challenging and improving my ideas. This endeavor would not have been possible without you. Your invaluable advice has contributed greatly to my academic and personal development.

I would like to express my deepest gratitude to Prof. Olaf Zukunft for awakening my interest in research and encouraging me to pursue a Ph.D. Your encouragement and expertise contributed significantly to the successful completion of this thesis. Thank you for always having time for me and for your continued dedication to my academic and personal development. Your unwavering support has been an anchor and will always be remembered.

I would also like to thank Prof. Marina Tropmann-Frick for her support and inspiring advice, especially in the early stages of my Ph.D. It was a pleasure to co-author a paper with you.

I have had the pleasure of working with many colleagues and friends who have supported me along the way. I have had many inspiring conversations with you, but also spent many enjoyable hours outside of research. Without your company, this experience would have been far less enriching.

Finally, words cannot express my gratitude to my family for their unconditional support throughout this journey.

Abstract

Machine Learning (ML)-based text classification offers a promising and scalable approach to automate the classification of text, such as app reviews or social media posts. However, the applicability of text classifiers in real-world settings is limited by inherent uncertainties and potential performance gaps.

This thesis explores the potential of the emerging field of Human-in-the-Loop (HiL) to address the applicability challenges for ML-based text classification. Our goal is to increase applicability by efficiently incorporating humans into the text classification pipeline. To this end, we review the current literature on HiL and develop a pattern catalog to provide software developers with best practices for designing HiL systems. Our catalog aims to facilitate the development and deployment of applicable HiL systems that integrate humans into the ML loop.

Next, this thesis proposes and evaluates novel frameworks that implement three HiL patterns. Addressing common applicability challenges such as high computational resource requirements, limited classification performance, and high model latency. In particular, the proposed frameworks utilize the concept of prediction uncertainty to coordinate human efforts. We evaluated the effectiveness of our frameworks using datasets from the domains of software engineering, online journalism, and social media analysis. Our contributions enable more applicable solutions for the cost-optimized training and deployment of text classifiers.

Furthermore, we propose a novel framework for explaining the prediction uncertainty of text classifiers in order to improve user understanding of classification decisions. While existing explanation techniques mainly explain the provided class label, we build on the concept of prediction uncertainty and make it explicit to users.

Finally, based on our HiL patterns, this thesis develops REM, a visual tool for the real-time moderation of online forums. REM enables a semi-automated moderation of continuous streams of user comments by integrating various HiL patterns and HiL collaboration mechanisms. Experiments with REM show promising results, achieving a significant increase in classification performance (from 78.48% to 96.08%) while requiring manual moderation of only 25% of

the data. These contributions empower domain experts to be efficiently involved in the text classification process, ultimately improving the applicability and efficiency of forum moderation in real-world settings. REM was developed specifically for online journalism, but can be easily adapted to other domains.

Kurzfassung

Textklassifikationsverfahren, die auf maschinellem Lernen basieren, sind ein vielversprechender und hochgradig skalierbarer Ansatz, um die Klassifikation von Texten wie beispielsweise App-Rezensionen oder Kommentaren in sozialen Medien zu automatisieren. Die Anwendbarkeit solcher Verfahren wird jedoch durch inhärente Unsicherheiten und potentielle Leistungsdefizite in Bezug auf die Anforderungen, die in realen Anwendungsgebieten auftreten, eingeschränkt.

Diese Arbeit untersucht die Eignung des aufstrebenden Human-in-the-Loop-Ansatzes, um die Anwendbarkeit maschineller Lernverfahren für die Textklassifikation zu adressieren. Unser Ziel ist es, die Anwendbarkeit durch die effiziente Integration des Menschen in den Klassifikationsprozess zu verbessern. Dazu wird zunächst ein Überblick über den aktuellen Stand der Literatur zum HiL-Ansatz gegeben. Ein Katalog von Entwurfsmustern wird entwickelt, um Softwareentwicklern bewährte Praktiken für den Entwurf von HiL-Systemen bereitzustellen. Die Muster können den Entwurf und die Inbetriebnahme von HiL Systemen erleichtern.

Anschließend werden neuartige Rahmenwerke für die Umsetzung von drei HiL-Mustern vorgeschlagen und evaluiert. Diese adressieren allgemeine Herausforderungen der Anwendbarkeit von Textklassifikatoren, wie z.B. den enormen Bedarf an Rechenressourcen, eine begrenzte Klassifikationsgenauigkeit und eine hohe algorithmische Latenz zwischen menschlichen Interaktionen während des Trainingsprozesses. Die vorgeschlagenen Implementierungen nutzen insbesondere das Konzept der Vorhersageunsicherheit, um die Effizienz menschlicher Eingriffe zu optimieren. Die Effektivität dieser Ansätze wird anhand von Datensätzen aus der Softwareentwicklung, dem Online-Journalismus und der Analyse sozialer Medien evaluiert. Unsere Beiträge ermöglichen anwendbarere Lösungen für das kostenoptimierte Training und den Einsatz von Textklassifikatoren.

Darüber hinaus schlagen wir ein neues Rahmenwerk zur Erklärung der Vorhersageunsicherheit von Textklassifikationsergebnissen vor, um das Verständnis der Klassifikationsergebnisse für den Benutzer zu verbessern. Während bestehende Ansätze hauptsächlich das Klassifikationsergebnis erklären, bauen wir auf dem Konzept der Vorhersageunsicherheit auf und kommunizieren diese dem Nutzer.

Basierend auf unseren HiL-Mustern entwickelt diese Arbeit REM, ein visuelles Werkzeug für die Echtzeit-Moderation von Online-Foren. REM ermöglicht die halbautomatische Moderation kontinuierlicher Ströme von Nutzerfeedback durch die Integration verschiedener HiL-Muster und HiL-Kollaborationsmechanismen. Experimente mit REM zeigen vielversprechende Ergebnisse mit einer signifikanten Erhöhung der Klassifikationsgenauigkeit (von 78,48% auf 96,08%), während nur 25% der Daten manuell moderiert werden müssen. REM ermöglicht die effiziente Einbindung von Moderatoren in den Textklassifikationsprozess, was letztlich die Anwendbarkeit und Effizienz von Forenmoderation in realen Anwendungsfällen verbessert. REM wurde speziell für den Bereich des Online-Journalismus entwickelt, kann aber leicht an andere Bereiche angepasst werden.

Contents

Chapter 1

Introduction

This chapter presents the problem statement, goals, and unique contributions of the thesis, as well as its scope. The chapter concludes with an overview of the general structure of the thesis.

1.1 Problem Statement

User-generated content written in natural language – commonly referred to as **user comments** – has become a fundamental component of many online platforms, including marketplaces, news sites, and social media. Users typically participate by posting and reading comments about products or services. User comments are recognized as a valuable and desired form of feedback that provides a rich and insightful source of real-world opinions and experiences [234]. In addition, user comments are increasingly the target of misuse, such as hate speech and spam [111]. Across various domains, there is a strong need to extract and leverage the hidden insights embedded in large collections of user comments while implementing content filtering strategies [234, 283, 285]. The most common type of analysis is text classification [221], which aims to group comments into existing classes, such as relevant and irrelevant.

Text is qualitative in nature and represents unstructured data. While natural language text follows grammatical and syntactical rules, it does not specify how the information it contains is organized. Natural language text is inherently flexible and inconsistent, making it challenging to analyze [267]. There are two main approaches to text classification. The first approach is **manual** text classification. This requires human annotators to carefully read and think about each individual text. However, as the volume of data increases, manual text classification becomes increasingly impractical. Manually classifying large amounts of text is labor-intensive, repetitive, tedious, and highly vulnerable to unintentional errors [8]. For example, when presented with an overwhelming amount of information, it can be difficult for humans to extract specific useful parts. The second approach is **automated** text classification. With the use of sophisticated algorithms, we can automatically classify textual data in many ways. Automated text classification makes information retrieval faster and more efficient [141]. It alleviates the problem of information overload by eliminating the subjectivity of manual classification. This thesis focuses on automated text classification using Machine Learning (ML) models.

Natural language text is inherently flexible and inconsistent. Automated text classification, especially using ML, faces a variety of uncertainties that can affect its reliability and classification performance [87]. As a result, practitioners typically have to deal with a significant amount of misclassification. The development of strong text classifiers has become a core task in ML research and has received much attention [88, 195]. ML-based text classification is a difficult task due to critical challenges faced by practitioners, which we outline below.

Lack of Applicability. There is typically a gap between the capabilities of ML-based systems and the actual needs of their users and application domains [242, 282]. The applicability of ML systems generally refers to their quality and alignment with critical requirements within a specific use-case. Examples include meeting computational constraints [64], achieving appropriate performance metrics that satisfy the goals and needs of the application domain [183], and meeting fairness-related criteria [250]. Additionally, an ML system is considerably applicable if it achieves desired outcomes under real-world conditions or aligning with established user experience guidelines [361]. However, as outlined by Tambon et al. [356], it is difficult to guarantee or even certify the safety or correctness of ML models. Many dimensions of applicability, such as accuracy and robustness, are also highly subjective and difficult to quantify [242, 376]. Current ML models are prone to uncertainty [87] and often exhibit limited classification performance [73] coupled with a general lack of transparency [136]. This thesis aims to develop solutions that improve the applicability of text classification systems. Specifically, it focuses on providing highly accurate classification outcomes tailored to better fit the requirements of a given domain, conserving human and computational resources, and ensuring high user experience and productivity in terms of model latency.

Missing Consideration of Humans in the ML Loop. The Human-in-the-Loop (HiL) approach [155] represents a new direction for ML systems, emphasizing the tight integration of humans as a critical part of the ML pipeline. HiL holds great promise for improving the applicability of text classification systems [155, 221]. It embodies the concept of harnessing the strengths of humans and

ML models by allowing humans to provide feedback to an ML model when it faces challenges in providing appropriate outcomes or requires external feedback. However, the suitability and effectiveness of HiL still need to be researched. The research landscape on HiL approaches is somewhat convoluted. From a software engineering perspective, there is a lack of consensus on how to effectively pair humans and ML models, e.g., to achieve superior text classification performance while maintaining the cost-effectiveness of human involvement. How HiL can mitigate the common applicability challenges of current ML-based text classifiers remains largely unknown. Exploring the potential of HiL to address these shortcomings is an essential but under-researched area.

Missing Tool Support. Analyzing, filtering, and aggregating user comments requires robust tool support, which is recognized as a significant challenge in various domains dealing with large volumes of user comments [90, 283]. There is an urgent need for tool support to manage the flood of information and strategically allocate human effort to support the text classification process [234, 285]. Dealing with an increasing number of diverse user comments is challenging. Typically, user comments are monitored to gain and extract insight into what users are expressing, or to filter out unwanted content such as hate speech or spam. Ensuring high classification performance during operation is one of the most important requirements for ML systems to be effective [133].

1.2 Objectives and Contributions

The goal of this thesis is to investigate frameworks for improving the applicability of ML-based text classifiers in real-world settings using HiL. It attempts to overcome the significant applicability challenges of ML-based text classification models by incorporating feedback from human domain experts into the classification process. A critical aspect of this endeavor is the consideration of applicability factors such as human-resource efficiency, latency in terms of user experience and transparency, and technical considerations such as computational resource consumption.

First, this thesis analyzes the challenges associated with ML-based text classification models and investigates the HiL approach to address some of the outlined challenges. It reviews the literature on existing HiL approaches and develops a catalog of HiL design patterns that involve human intervention to improve the applicability of text classification models. Second, we investigate and propose concrete frameworks to implement three prevailing HiL patterns. In addition, this thesis introduces one of the first frameworks to explain prediction uncertainty in text classification. Third, this thesis develops REM, a functional prototype for human-resource-aware moderation of online forums in real-time. REM assists domain experts in accurately moderating user comments in online forums. We briefly summarize the main contributions of our work:

Human-in-the-Loop Pattern Catalog. Chapter 4 proposes a catalog of 12 HiL patterns to guide software engineers in developing HiL systems. The research landscape of HiL is rather convoluted, with a notable lack of best practices for integrating and leveraging human effort within the ML loop to achieve specific goals. Design knowledge is essential because careful consideration of human effort is critical in HiL systems, especially given their limited availability and high cost. Most existing HiL systems focus primarily on creating the initial training dataset for model training, while significant challenges arise during the deployment of ML systems. The catalog is constructed through a literature review and our experience in developing HiL systems. The emphasis lies on human domain experts and the task of labeling. We distinguish between training and operation patterns. Training patterns are primarily concerned with the effective collection of training data or model tuning during the training phase, while operation patterns apply post-training, i.e., when the model is operationalized.

More Applicable Text Classification via HiL. We present concrete frameworks for implementing selected patterns from our HiL pattern catalog to improve the applicability of text classifiers in real-world scenarios. The focus is on achieving strong and practical classification performance, a primary goal of text classification. Furthermore, we aim to increase the applicability of text classification under constraints such as limited computational resources and resource-efficient use of human effort. In addition, we explore a framework to maintain high usability when humans interact with ML models during training by minimizing model latency.

- **Computational-aware Active Moderation.** Chapter 5 explores a novel uncertainty-based HiL framework for improving the classification performance of text classifiers in computationally resource-constrained, low-end infrastructures. It addresses the challenge that state-of-the-art classifiers are often highly complex, expensive to train and deploy, and may not be readily applicable in certain scenarios. In such cases, lightweight classifiers must be used. However, these typically have lower classification performance. We find that a HiL approach, in which the most uncertain predictions encountered during operation are referred to humans, can significantly improve classification performance at the expense of human effort. Even in the presence of human error, superior results were obtained compared to purely manual or purely ML-based classification while relying on lightweight classifiers. Hence, strong classification performance can be extended to a broader audience.

- **Human-resource-aware Active Moderation.** Chapter 6 develops a framework for optimizing the amount of human effort needed to correct highly uncertain predictions during operation. Human resources are cost-intensive and cannot be allocated arbitrarily. Therefore, this thesis introduces a human-resource-aware HiL framework that aims to maximize the classification performance of a classifier during operation while minimizing human effort. The proposed saturation-based moderation strategy provides a solution to the trade-off between classification performance and human effort. Our results show that with our novel framework, practitioners can achieve accuracies of up to 99% with reasonable human effort. Minimizing human effort while maximizing classification performance increases applicability. Consequently, human-resource-aware Active Moderation allows users to spend their time on more interesting and less redundant tasks other than labeling.

- **Proxy-based Active Learning.** Chapter 7 explores a time-efficient framework for Active Learning, a HiL pattern that aims to minimize the number of labels a human must provide to train well-performing models. In Active Learning, typically highly complex models must be re-trained between label assignments, often resulting in enormous waiting times. To address this challenge, we propose a Proxy-based Active Learning framework for text classification. The goal is to reduce model latency between label assignments while leveraging the high classification performance of complex state-of-the-art classifiers during deployment. Our results show that our framework can maintain a latency of less than 1 second while providing training data that can be effectively reused to train state-of-the-art ML models.

BayLUXT: Bayesian Local Uncertainty Explanations. Chapter 8 introduces BayLUXT, a model-specific local explanation framework designed to explain prediction uncertainties in text classification. In total, BayLUXT provides five types of explanations, including word-level and sequence-level uncertainty explanations. While uncertainty estimates offer valuable insights when an ML model fails to provide reliable outcomes, they remain intransparent and lack user trust. BayLUXT is one of the first frameworks to explain why predictions were deemed uncertain by a text classifier. We have conducted several experiments that demonstrate the effectiveness of our framework and a human evaluation that underscores its significance. Our results suggest that practitioners benefit from uncertainty explanations because they offer valuable insights into a model's decision-making process that traditional explanation techniques do not provide. For instance, humans can understand which words in a text contributed most to a highly uncertain prediction.

REM: A Prototype for the Real-Time Moderation of Online Forums. Chapter 9 introduces REM, a functional prototype designed for the real-time moderation of large-scale online forums. REM aims to efficiently allocate human effort to achieve a desired level of classification performance. The tool integrates several HiL patterns and our frameworks, including those introduced in the Chapters 5, 6, and 8, while relying heavily on visual interactive interfaces. REM directly prompts domain experts to label instances marked as highly uncertain, while enabling exploratory analysis of online forums. With REM, domain experts are not only involved in labeling, but also gain additional insight and knowledge about their data. The architecture of REM is based on several big data frameworks that ensure its scalability and enable real-time processing capabilities.

1.3 Scope

We briefly outline the scope of this thesis.

Domain and Use-case Selection. This thesis focuses mainly on the domains of software engineering, online journalism, and social media analytics. User feedback has emerged as a critical aspect enabling data-driven decision-making in **software engineering** practice. Especially in app stores and marketplaces, user feedback is a valuable source of information about how users feel about apps and services. From a software engineering perspective, mining bug reports, feature requests, or praise hidden in user comments provides highly desired information for software maintenance and evolution. Online forums have become an integral part of **online journalism**, offering valuable feedback and serving as a desired communication channel for audiences. Harnessing their constructive value and filtering out harmful content from the growing flood of user comments is a critical challenge. **Social Media Analytics** has become an indispensable tool for gaining insight into the behavior, opinions, and trends of millions of users. By analyzing user comments on social media sites such as Twitter, stakeholders can gain insight into how audiences respond to specific content, identify emerging trends, and determine what topics are trending and how users feel about them.

Text-based User Comments. The scope of this thesis focuses on text-based user comments as the data to be analyzed. This content typically encompasses any text created and shared by users on online platforms, social media, forums, blogs, or other digital platforms where users contribute or share content. In addition, we limit our analysis to text written in the English language. This limitation helps to maintain consistency and facilitates the analysis process by narrowing the scope to a specific language.

Domain Expert Feedback. Domain expert feedback encompasses the opinions, insights, and suggestions offered by individuals with expertise or specialized knowledge in a particular field or domain. These experts possess a profound understanding, experience, and knowledge of their respective domains, enabling them to provide valuable perspectives, critiques, and recommendations on specific topics, issues, or projects. Domain experts are usually non-ML experts. They often lack technical knowledge about how ML models work or how they are developed. From a HiL perspective, feedback from domain experts is most promising because it has the potential to provide valuable knowledge about the domains or tasks that an ML model does not process. This knowledge can help an ML model in cases where it does not perform well on its own.

1.4 Structure

The remainder of this thesis is structured in three parts:

Part I – Problem. Chapter 2 outlines the fundamental concept of ML-based text classification. It emphasizes the concept of prediction uncertainty, which we will rely on heavily in the rest of this thesis. It then outlines the real-world use-cases of text classification that we will focus on. Chapter 3, analyzes and highlights the general challenges of ML-based text classifiers that limit their applicability, especially in real-world settings.

Part II – Solution. Chapter 4 introduces the HiL approach, which we further adapt to overcome some of the identified challenges to enable more applicable text classification systems. It reviews the literature on existing HiL solutions and provides a catalog of HiL patterns describing best practices.

In the following chapters, we propose three HiL frameworks to address critical operational and training-related challenges. For each chapter, this thesis outlines how domain experts in the fields of online journalism, software engineering, and social media analytics can use the HiL approach to achieve more applicable text classification. Chapter 5 investigates a HiL framework to improve the classification performance of lightweight classifiers in computationally constrained environments. Chapter 6 investigates a human-resource-aware HiL framework that aims to minimize human effort while maximizing the classification performance during operation. Chapter 7 adapts a Proxy-based Active Learning framework for text classification to improve the user experience during training. Finally, Chapter 8 proposes BayLUXT - a novel framework to explain the prediction uncertainties of a text classifier.

Part III – Synopsis. Chapter 9 describes the engineering and development of REM, a functional prototype for the effective moderation of user comments. It outlines REM's requirements, describes its architecture, and discusses its HiL workflow and user interface. Chapter 10 summarizes our contributions throughout the thesis, discusses threats to validity, and highlights further work.

Part I

Problem

Chapter 2

Foundations

This chapter introduces the basic concepts of this thesis. The focus is on aspects that are central to understanding the HiL patterns and frameworks presented in the following chapters. Section 2.1 gives a brief introduction to the ML-based classification of natural language text. It discusses the traditional classification pipeline and its subtasks. Section 2.2 introduces the concept of classification uncertainty as a first step towards more applicable text classification. It defines various sources, types, and techniques for modeling and quantifying prediction uncertainty.

2.1 Text Classification

Text classification [221] is one of the most fundamental and essential tasks in natural language processing (NLP) [264]. This section formally defines the text classification task, provides an overview of the text classification pipeline, discusses common classification models, and describes the validation process.

2.1.1 Overview

Natural language text is a rich and desirable source of information [85, 234, 240, 263, 348, 368]. It is one of the most common types of user-generated data. Most natural language text today is created and distributed over the Internet. Typically, human users communicate by sharing and responding to other content, for example via email [53], within online forums [234, 348], marketplaces [240, 247] or support requests [179, 368]. Such text messages can contain valuable insights such as aspects, sentiments, personal opinions, or feedback [234] that are of great interest to many application areas [53, 60, 85, 141, 240, 283]. Automating the extraction of insights from large amounts of textual data is desirable. A high degree of automation is promising because it saves much time that would otherwise be spent by humans [141]. The most common task of natural language processing is *text classification* [221].

Text classification is the process of assigning text instances to predefined classes [165, 221, 264]. For example, an email provider commonly offers its customers protection against spam [53, 364], such as undesired advertisements. For this purpose, incoming emails must be classified into "spam" and "non-spam" in order to filter out the latter. Text classification is a popular research area with high practical impact. It has numerous real-world applications, including monitoring [234], content filtering [283, 285], and indexing [240]. Effective text classification can potentially assist practitioners across various domains in their daily workflows [234, 285]. Text classification is critical for organizing relevant text and facilitating data-driven decision-making [241]. For instance, within the field of software engineering, text classification has shown high potential, including the classification of issue links [237], facilitating onboarding in open-source projects by classifying bug reports into beginner and expert levels [352], and identifying knowledge types in API documentation [112]. However, analyzing natural language text is challenging due to its complexity and individual nuances, making it both challenging and resource-intensive in terms of human effort.

2.1.2 Problem Specification

Let X be a set of input texts $x \in X$ and $Y = \{1, ..., C\}$ a set of $C \geq 2$ classes, also called labels. Text classification [62] deals with a set of input-output pairs $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\} \subset X \times Y$, where we assume an unknown relationship between each pair (x, y). Given an input x, classification aims to retrieve its true label $\hat{y} \in Y$. Text classification is part of supervised ML [199], which forms a family of ML models characterized by the way they acquire knowledge. A classification model acquires knowledge about the unknown input-output relationship $X \times Y$ from a set of already labeled examples $D = \{(x_i, \hat{y}_i)\}_{i=1}^{N} \subset X \times Y$ of size N to derive the labels Y of inputs X. To better distinguish between true and inferred labels, we use $\hat{y}$ to refer to the true label (ground truth) of x and y as the artificially inferred label of x, which may differ from $\hat{y}$.

Formally, an automated text classifier can be described as a function $f^{\omega} : X \rightarrow Y$ that maps an input text to a finite set of outputs, where ω are some function weights. The input-output mapping $\hat{y} \approx f^{\omega}(x)$ is learned in a process called training. Training usually involves finding the optimal weights $\hat{\omega} \in \Omega$ of f that minimize a loss metric L [257]:

$$\hat{\omega} = \operatorname*{argmin}_{\omega \in \Omega} \frac{1}{N} \sum_{i=1}^{N} L(\hat{y}_i, x_i; f^{\omega}) \tag{2.1}$$

Most modern classification models provide class probabilities $p(y = c|x, D)$ that describe the relationship between X and Y. The probabilities indicate the likelihood that a given input x belongs to a particular class c under the control of data D. The class of a new input x is derived by selecting the class that receives the highest conditional class probability, that is:

$$y = f(x) = \underset{k \in \{1,\ldots,C\}}{\text{argmax}}\; p(y = k|x, D) \tag{2.2}$$

To build a text classifier, an initial set of correctly labeled examples $D_{all} = \{x_i, \hat{y}_i\}_{i=1}^{N} \subset X \times Y$ is needed. Examples $(x, \hat{y}) \in D_{all}$ define how a classifier f should behave given an unseen input x. When training a classifier ,D_{all} is divided into a training set $D = D_{train}$, a validation set D_{val}, and a test set D_{test} with $D_{all} = D_{train} \cup D_{test} \cup D_{val}$ and $D_{train} \cap D_{test} \cap D_{val} = \emptyset$. The different quantities serve the following purposes:

- The **training data** $D = D_{train} = \{(x_i, y_i)\}_{i=1}^{|D_{train}|}$ is a sample of D_{all} that is used to train the model. A model learns the relationship between X and Y from the training data.
- **Validation data** $D_{val} = \{(x_i, y_i)\}_{i=|D_{train}|+1}^{|D_{train} \cup D_{val}|}$ is an additional data segment that injects new data into the training process. Validation data is used to evaluate and fine-tune the model and prevent common learning issues such as overfitting [230]. While only some models can be tuned or adapted with additional data during training, validation data is only occasionally needed.
- The **test data** $D_{test} = \{(x_i, y_i)\}_{i=|D_{train} \cup D_{val}|+1}^{|D_{all}|}$ is used to assess whether a trained model makes useful prediction after the training process. Its purpose is to check and confirm how well the model performs on unseen data before deploying the classifier in a real-world scenario. Test data is needed because the overall classification performance of a classifier can only be assessed empirically.

This thesis focuses on the traditional classification setting, where each input belongs exclusively to one and only one of multiple non-overlapping and immutable classes. Classification is called *binary* when there are only two classes ($C = 2$) and *multi-class* when there are more than two classes ($C > 2$).

2.1.3 Text Classification Pipeline

Feature extraction [189] and classification models [255] form the technical foundation of automated text classification. However, text classification involves not only the inference step of a classification model, but also upstream and downstream tasks such as data cleaning, classifier validation, and deployment [224]. Text consists of sequences of characters that cannot be fed directly into ML models. This uniqueness has led to the development of various NLP techniques to facilitate the use of text for ML models [192, 264]. In general, the application of text classification models follows a generic, domain-independent ML pipeline, as depicted in Figure 2.1.

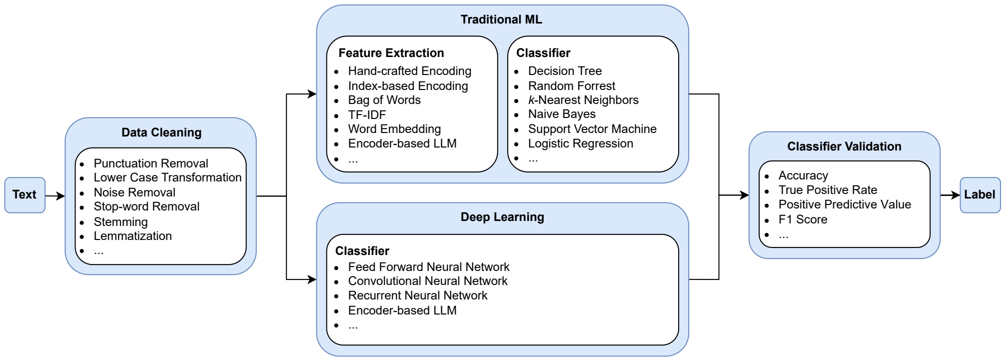


Figure 2.1: The generic text classification pipeline illustrates the process of transforming input text into a classification label. The pipeline shows the workflow using both traditional and deep learning approaches. Established implementations are listed for each step. Adapted from Li et al. [224].

The pipeline illustrates the consecutive steps that data takes from raw text to the final class label. These are data cleaning, feature extraction, classifier selection, validation, and the deployment of the classification label [24, 37, 51, 200, 224]. The trend is towards deep learning [213] models, which learn an optimal feature representation in addition to the actual classification task [265]. Feature extraction is replaced by autonomously learned, model-internal feature representations, eliminating the need for manual feature selection and engineering. In comparison, traditional ML classifiers require the selection of an appropriate feature extraction technique.

Because each text classification task is unique, there is no general solution for implementing each step of the pipeline. The overall efficiency of a text classifier depends heavily on its downstream task. However, numerous best practices and design patterns [108, 137, 146, 209, 385] have been proposed that are likely to yield high-quality and well-performing text classifiers. The most common goal of the text classification pipeline is typically to obtain the most accurate classifier for deployment [213]. The following section briefly describes each generic step of the text classification pipeline and outlines best practices.

2.1.4 Data Cleaning

Like any raw data, text needs to be pre-processed, i.e., cleaned, to improve its quality. ML models rely heavily on expressive and meaningful input to maximize analysis outcomes, which requires high-quality data. Data cleaning aims to make text instances more predictable and analyzable, i.e., more accurate, reliable, and consistent [299].

A fundamental challenge in text classification is the high dimensionality of the feature space. The feature space in NLP models often consists of words or tokens, which tend to be very large. Natural language is inherently complex and diverse, characterized by large vocabularies, diverse grammatical structures, and nuanced semantics. Such large and complex feature spaces contribute to high computational complexity and a more challenging training process. An illustrative challenge is overfitting [230], a common issue where the model effectively fits the training data but exhibits a poor classification performance when applied to other datasets.

Data cleaning techniques for text focus on reducing the dimensionality of the feature space by removing obsolete characters and words from the input [269]. The fundamental concept of text cleaning is to standardize text instances and remove ambiguous, noisy, and irrelevant terms that are unlikely to help distinguish between classes. Data cleaning aims to simplify the classification.

Data cleaning is domain- and task-specific because the meaning of words and characters can vary depending on the context. Data cleaning should be done carefully, as it can add noise or disrupt the semantic integrity of text [240]. Poorly chosen cleaning steps can have a negative impact on the classification performance. The following section outlines common cleaning techniques for natural language text. It focuses on general best practices that have proven very successful for text classification [86, 200, 267, 355, 375].

Punctuation Removal. Punctuation marks such as ".", ",", ":", ";", "!", or "?" generally do not provide much information for classification tasks and are therefore usually removed from the input text, such as:

"Does he drink tea?" $\Rightarrow$ "Does he drink tea"

Lower Case Transformation. NLP techniques typically distinguish between uppercase and lowercase letters. Differentiating between mixed-case words is often unnecessary, as this information rarely adds additional meaning given the semantic equivalence of the terms, e.g.:

"Does he drink tea?" $\Rightarrow$ "does he drink tea?"

Noise Removal. Text obtained from web crawling may contain special characters, metadata, and markup instructions such as HTML tags. These are considered noise because they can introduce unwanted inconsistencies and ambiguities during training, while having little or no value for the actual classification task. Noise removal involves removing such noisy content from text instances. An example of noise removal is demonstrated below:

“Does he drink tea?” ⇒ “Does he drink tea?”

Stop-word Removal. Stop words are typically high-frequency words within a language. They provide little to no information as they do not contribute to the true meaning of a text. Removing them helps the model focus on meaningful and important words. Common examples of stop words are “*a*”, “*that*”, “*from*”, and “*while*”. The cleaning step might look as follows:

“Does he drink tea? ⇒ drink tea?”

Stemming and Lemmatization. For grammatical reasons, words with related meanings exist in different forms, e.g., “*study*”, “*studies*”, “*studying*”. Stemming and lemmatization are techniques that aim to replace related word forms with a single, canonical representative. Stemming identifies a common root form of a word by removing or replacing predefined suffixes such as “*-ing*”, “*-s*”, “*-ed*” from words. The resulting stem is not necessarily a valid word, but stemming can be done quickly. In contrast, lemmatization considers the contextual meaning of a word and maps it to its dictionary base form, also known as a lemma. The lemma is a valid word; for example, “*better*” can be lemmatized to “*well*”. A lemma is identical for all its inflectional forms, e.g.:

“Changing the system made it better.” ⇒ “Change the system make it well.”

For a comprehensive overview of data cleaning techniques for NLP, see Tabassum and Patil [355] or Naseem et al. [267].

2.1.5 Feature Extraction

Classification models based on ML require meaningful numerical feature vectors as input to function. Encoding or embedding is the process of transforming textual data into numerical feature vectors. The effectiveness of classifiers is largely limited by the feature representations, so creating meaningful encodings is an essential task of NLP. The most commonly used techniques are as follows [177, 200, 227]:

Hand-crafted Encoding. Hand-crafted encodings are manually selected or defined features derived from the text or associated metadata. Manually engineering feature vectors is challenging and time-consuming, requiring a deep understanding of the problem and much analysis, experimentation, and testing. A typical example is the length of a text and context-specific metadata, such as the number of texts posted by the same author or the time-frequency of text postings [247]. However, hand-crafted features do not translate well to other domains. Purely hand-crafted text encodings are of little relevance in modern text classification, as they fail to account for syntactic and semantic features.

Index-based Encoding. The most basic technique for representing text numerically is to represent the occurrence of a word in a text by a unique number, also known as an index. First, an invariant vocabulary is created that assigns a unique index to each word in a given text corpus. Then, a vocabulary is formed in which a word is associated with a corresponding index. An index-based encoding involves replacing each word with its corresponding index from the vocabulary. The vocabulary size is a critical design consideration, as it affects computational and storage costs by determining the number of words to be processed. A vocabulary that is too small can result in a large number of unknown words, i.e. words that cannot be indexed because they are not in the vocabulary. An index-based encoding might appear as follows:

$$\text{“Does he drink tea?”} \Rightarrow [1,\ 6,\ 5,\ 3,\ 14] \in \mathbb{Z}^{|x|}$$

Bag of Words (BoW). A BoW feature vector employs the frequency of words within a text as its representation [200]. Analogous to index-based encoding, a vocabulary is constructed to encode each word of a given text corpus. Each index of a BoW vector represents the frequency of the corresponding word in a text. A value of 0 indicates the absence of that particular word. Consequently, the length (n) of the feature vectors must match that of the vocabulary, making the model much more complex and increases its computational complexity. BoW representations capture only the number of word occurrences; they do not address word order or reflect the semantic relationship between words.

$$\text{“Does he drink tea?”} \Rightarrow [0,\ 0,\ 1,\ 1,\ 0,\ ...,\ 0,\ 0,\ 1] \in \{0, 1\}^n$$

TF-IDF. A significant limitation of BoW is that all words are equally weighted in the final feature vector. However, some words may be more relevant for a classification decision. TF-IDF (*Term Frequency-Inverse Document Frequency*) is a technique that weights a word in relation to a text and the entire corpus

[346]. TF-IDF measures the frequency and distinctiveness of each word in a corpus. Words that occur frequently throughout the corpus are considered less important, while less frequent words are deemed more salient and receive a higher weight.

The *Term Frequency* (TF), as defined in Eq. 2.3, is calculated as the number of occurrences of a word w within a text d, denoted as $\text{count}(w, d)$ divided by the total number of words in d, which is $\sum_{w' \in d} \text{count}(w', d)$.

$$TF(w, d) := \frac{\text{count}(w, d)}{\sum_{v \in d} \text{count}(v, d)} \tag{2.3}$$

The *Inverse Document Frequency* (IDF) is a measure of the information content of a word within a corpus. It can be used to assess whether a word is discriminative or common within a corpus. The IDF is defined as the logarithmically scaled fraction of texts containing a word, that is:

$$IDF(w, D) := \log \frac{|D|}{|\{d \in D | w \in d\}|} \tag{2.4}$$

The $TF\text{-}IDF$ representation is then calculated as the product of the term frequency (TF) and the inverse document frequency (IDF).

$$TF\text{-}IDF(w, d, D) := TF(w, d) \cdot IDF(w, D) \tag{2.5}$$

A $TF\text{-}IDF$ vector may look as follows:

"Does he drink tea ?" $\Rightarrow$ [0, 0.62, 0.12, 0, 0, ..., 0, 0, 0.29] $\in \mathbb{R}^n$

Word Embeddings. Feature representations based on word frequency do not take into account the actual meaning of a word. As introduced by Hinton [151], word embeddings attempt to overcome this limitation. They constitute a family of unsupervised learned text representations that capture the meaning of words from the context in which they are used [42, 253, 287]. Word embeddings encode both the semantic and syntactic information of a word in a vector space [322]. A high-dimensional dense vector represents each word in such a way that the cosine similarity between two words captures their semantic relationship. Words that are closer in the vector space are assumed to be semantically similar, while distant words are assumed to be semantically dissimilar.

Word embeddings allow for simple algebraic operations, demonstrating their representational accuracy. Mikolov et al. [252] have demonstrated the expressive power of word embeddings across various word similarity tasks. Let *vector(w)* denote a function that returns a word embedding of the word w. First, they show that the expression "*vector(King) - vector(Man) + vector(Woman)*" yields a vector that is closest to the word *Queen*. Second, they show that the embedding space can also reflect other relationships, such as the degree of comparison. For example, the word "*big*" is shown to be similar to "*bigger*" in the same sense that "*small*" is similar to "*smaller*".

Word embeddings are typically pre-trained on massive text corpora and can be further refined on domain-specific data. Word embeddings map each unique word to its corresponding vector space representation. Thus, a unique word is always mapped to the same vector, regardless of its meaning. To illustrate this concept, consider the following sentences: *"I ate an apple"* and *"Apple announced a new line of phones"*. In these contexts, the word "apple" refers to a fruit in one context and to a technology company in the other. Word embeddings represent "apple" with a single vector, but this representation does not inherently distinguish between its different senses or meanings. Consequently, word embeddings cannot account for homonyms. When using word embeddings as text representations, each text is represented by a matrix $X \subset \mathbb{R}^{m \times d}$ where m is the number of processable words per text and d is the dimensionality of the word embeddings. A word embedding might look as follows:

$$vector(King) = [0.34, 0.12, 0.92, 0.03, 0.58, ..., 0.29, 0.64, 0.11] \in \mathbb{R}^d$$

Common word embedding models are word2vec [252, 253], GloVe [287], and fastText [42].

Large Language Models (LLMs). Recently, LLMs have attracted considerable attention due to their high empirical performance in distilling statistical meanings from language. This has been demonstrated by models such as BERT [88], RoBERTa [232], AlBERT [211], and DeBERTa [144], which were trained on extensive volumes of raw text using self-supervised learning techniques [371]. Current LLMs have been shown to provide state-of-the-art performance in several NLP tasks, including text classification [88].

Formally, a language model is a probability distribution over words [66]. They can be used to determine the probability of words occurring in a sequence $p(w_1, w_2, ..., w_n)$ or to estimate the probability of a word within a sequence of surrounding words $p(w_i | w_1, ..., w_{i-1}, w_{i+1}, ..., w_n)$. Language models are applied to identify the most likely words, phrases, and sentences based on what the model has learned.

Recent LLMs are primarily based on the Transformer [371], a deep learning architecture based solely on the attention mechanism [28]. The attention mechanism allows for modeling dependencies between words regardless of their distance in the input or output sequence [373]. A Transformer consists of an encoder and a decoder, each of which can be tailored for a specific purpose:

- **Encoder-based LLMs** consist of a Transformer encoder. These models take an input sentence and provide effective feature representations learned directly from extensive text collections. The encoder is trained to capture bidirectional contextual information from the input text, enabling it to create text encodings that encapsulate the overall meaning of the text. Recent models offer contextualized encodings for tokens, words, and entire texts. Contextualization means that the feature vectors capture the contextual meaning of words or tokens. Identical words or tokens with distinct meanings are represented by different features.

  Encoder-based LLMs are built in two phases: pre-training and fine-tuning [88]. During **pre-training**, the model is trained unsupervised on a very large collection of unlabeled and generic texts using a *Mask Language Modeling* (MLM) and *Next Sentence Prediction* (NSP) target. The objective of MLM is to randomly mask tokens from text sequences and let the model predict the original tokens based only on their context. NSP involves learning the relationships between pairs of texts. The model is asked to predict the relationship between two sentences, which may be logical, sequential, or random. The purpose of the initial training is to teach the model the basic syntax and semantics of words (general language understanding), i.e. how to use natural language correctly. Training aims to capture the essence of the language.

  Encoder-based LLMs should be **fine-tuned** to exploit their full potential and to reach state-of-the-art classification performance [354]. Without fine-tuning, the model will not know the meaning of domain-specific words and phrases, requiring updates using domain-specific data. The most famous encoder-based LLM is BERT [88] which stands for *Bidirectional Encoder Representations from Transformers*.

- **Decoder-based LLMs** consist only of a Transformer decoder. Given a sequence of words, a decoder-based LLM is trained to predict the next word in the sequence (MLM). In this setup, the LLM is capable of generating new texts. The text generation process involves computing probabilities for the next token, which is conditioned on the previously generated token. Users employ decoder-based LLMs to generate human-like text based on input instructions or prompts. Although decoder-based LLMs do not extract features from text input, they can be utilized to answer questions in an FAQ fashion. Text classification can be performed by transforming the classification task into a closed question problem [326].

Examples of decoder-based LLMs are the family of GPT (*Generative Pre-trained Transformer*) models [298] or the chatbot *ChatGPT* [279]. For a comprehensive review of LLMs, see Min et al. [254].

2.1.6 Classification Models

The subject of classification models is the concrete implementation of the prediction function $f(x)$, which infers the class membership y for a given input x. Many efficient text classification models have been developed and proposed over the past decades [255, 280, 390].

We can distinguish between traditional and modern deep learning models for text classification. While both provide the technical foundation to automate text classification, they have significant differences [168]. **Traditional ML** models are known for their simplicity and relatively low execution time. They generally require less training data to identify patterns, but are limited in their ability to capture complex and nonlinear correlations. These models utilize one-dimensional vector representations as input. Traditional ML models have a long history, with some models dating back to the 18th century. In contrast, **deep learning** models, which demonstrated practical usefulness around 1990 [214], are a subset of ML models that primarily use multiple complex stacks of computational layers to learn from large amounts of data. The most commonly used deep learning model is a *Neural Network* (NN) [142]. Although more complex and computationally intensive, deep learning enables more sophisticated decision-making, potentially outperforming traditional ML models [292]. However, deep learning models require a large amount of training data to recognize patterns and to generalize effectively to new data [317].

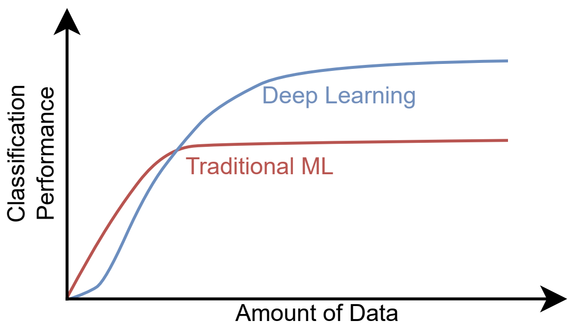


Figure 2.2: Expected classification performance of traditional ML compared to deep learning.

Figure 2.2 shows the approximate relationship between the amount of available training data and the expected classification performance for traditional ML and deep learning. This figure illustrates that traditional ML models tend to outperform deep learning when dealing with small datasets. However, as the amount of training data increases, deep learning is expected to perform significantly better.

This section briefly describes well-established ML models for automated text classification, covering both traditional ML and NNs for building deep learning models. Li et al. [224] offer a comprehensive survey of text classification models.

Traditional Machine Learning. Traditional ML provides a suite of fundamental models for text classification [177, 396]. Commonly applied traditional ML models for text classification include:

- **Decision Tree (DT).** A DT classifier is a tree-like structure of decision rules and leaf nodes [70]. A decision node has two or more branches representing the decision rules derived from the training data. For each classification, the tree is traversed according to the input. The leaf node reached defines the classification result.

- **Random Forrest (RF).** An RF classifier is an ensemble of DTs, each voting for a particular class outcome [48]. Each tree is built individually from a random sample of data. The results of all the DTs are combined into a single classification result. The final class prediction is the class that receives the most votes.

- ***k*-Nearest Neighbors (*k*NN).** A *k*NN is based on the assumption that the labeling of similar texts is equal [130]. Texts are classified according to the k most similar instances of the training dataset. A majority vote is taken to determine the final class result.

- **Naive Bayes (NB).** NB classifiers are a family of conditional probability models based on the assumption of feature independence [219]. NB classifiers use Bayes' rule to infer the conditional probability for each class outcome c. The class with the highest probability is taken as the final class outcome, that is:

$$y = f(x) = \underset{c \in \{1,\dots,C\}}{\operatorname{argmax}}\ p(c) \prod_{i=1}^{n} p(x_i|c) \tag{2.6}$$

  where x_i is the ith element of the feature vector x.

- **Support Vector Machine (SVM).** An SVM classifies data by finding an optimal linear hyperplane that separates features with a maximum margin [172]. The classification rule is based on which side of the hyperplane a data point occurs.

- **Logistic Regression (LR).** LR is a commonly used classification approach that can directly infer conditional class probabilities [96]. An LR model uses a *sigmoid* function to squeeze the output of a linear predictor function, denoted as $\theta_c x$, between 0 and 1, that is:

$$p(y = c|x) = \frac{\exp(\theta_c x)}{\sum_{k=1}^{C} \exp(\theta_k x)} \tag{2.7}$$

Neural Network (NN). An NN [142, 257] is the basic algorithm for constructing **deep learning** models. It operates as a biologically inspired supervised learning model comprising interconnected layers of nodes. In analogy to the human brain, the nodes are termed *neurons*. They are parameterized computational units that jointly perform summation and thresholding to optimize the classification result. Figure 2.3 shows the function of a single neuron.

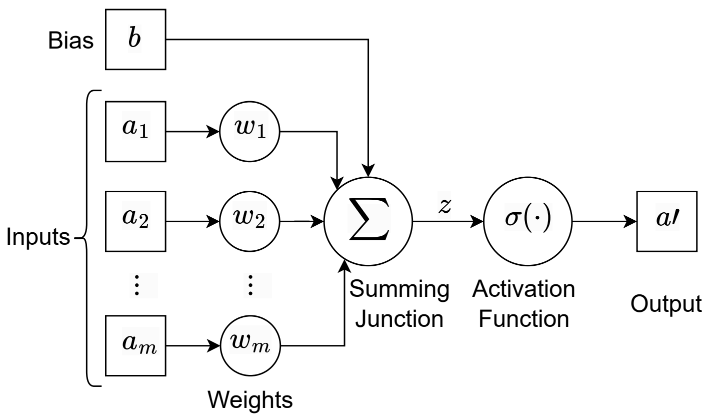


Figure 2.3: The processing steps of a neuron. Adopted from Haykin [142].

A neuron takes an input vector $a = (a_1, ..., a_m)$ of real numbers and maps it to a single output value a'. The input vector a is first multiplied by a weight vector $w = (w_1, ..., w_m)$, and the resulting weighted inputs are then summed up with a bias b. The bias enhances the computation by allowing a richer representation of the inputs by the model weights. The sum of the weighted inputs and the bias is defined in Eq. 2.8.

$$z = wa + b = \sum_k w_k a_k + b \tag{2.8}$$

Neurons should only activate (forward important information) under certain conditions. Therefore, z is passed through a typical non-linear activation function σ, which transforms a real number into a bounded output value, that is:

$$a' = \sigma(z) \tag{2.9}$$

A common activation function is the sigmoid function [135] which maps a real value to a value in the interval $[0, 1]$. The activation function σ acts as a threshold and determines whether the neuron should be active. A popular example is the *logistic function*, that is:

$$\sigma(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{2.10}$$

An NN consists of numerous connected neurons, where the output of one neuron is fed directly into other neurons. Some neurons are declared as input neurons and others as output neurons. The result of the output neurons is the class prediction. An NN computes a function $f^{\omega} : X \rightarrow Y$ based on its parameters $\omega = \{W^l, b^l | l = 1, ..., L\}$ consisting of the weights of each node $\omega^l_{ki} \in W^l$ and all biases $b^l_i \in b^l$. An NN is learned by tuning ω while minimizing a loss function. *Stochastic Gradient Descent* [316] solves the optimization problem. The gradient of the error function with respect to the weights and biases is computed iteratively. *Back-propagation* [318] is used to trace the loss through the network and update its weights and biases accordingly. Haykin [142] outlines the operation of gradient-decent and back-propagation in more detail.

Several types and variations of NNs have been successfully used for text classification [195, 208, 276]. The overall architecture, i.e., the structure, size, and type of neurons, is a critical factor in improving the classification performance of an NN. In the following, we briefly discuss the most commonly used variations of NNs. They differ mainly in the way information is processed and passed through the network.

- **Feed Forward Neural Network (FFNN).** An FFNN is the most basic type of NN. Neurons are arranged in successive layers, each receiving inputs from neurons in the preceding layer and transmitting outputs to the subsequent layer. An FFNN consists of an input layer that passes the input to a consecutive processing layer, one or more hidden layers, and an output layer that provides the classification results. An FFNN with only one or two hidden layers is commonly called a *shallow* NN due to its limited depth and relative simplicity. NNs with greater depth are categorized as deep learning models. Figure 2.4 depicts a fully connected 3-layer

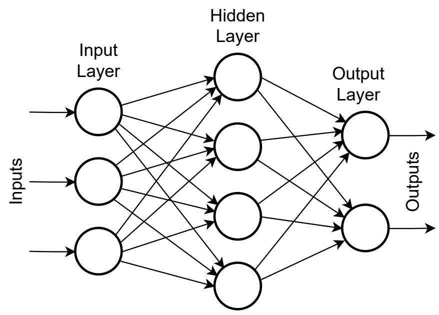


Figure 2.4: A 3-layer FFNN.

FFNN. Because of the fully connected layers, an FFNN interprets text as a bag of words that does not contain sequential information.

- **Convolutional Neural Network (CNN).** CNNs are used to learn features from two-dimensional data, such as word embedding matrices. They mainly consist of three different types of layers that are stacked on top of each other. These are *convolutional*, *pooling*, and *fully connected layers*, which serve the following purpose:
  - A **convolutional layer** consists of a set of filters whose parameters must be learned by training. Each filter is a matrix, usually smaller than the input. A filter convolutes with the input of the layer, creating an activation map. The filter slides over the height and width of the input, and the dot product of each element of the filter and the input window is computed. The result is a two-dimensional activation map. Figure 2.5 shows an example of the calculation of the first element of the activation map. Within a convolutional layer, a neuron is only connected to a local area of input neurons, thus reducing the dimensionality of the next layer.

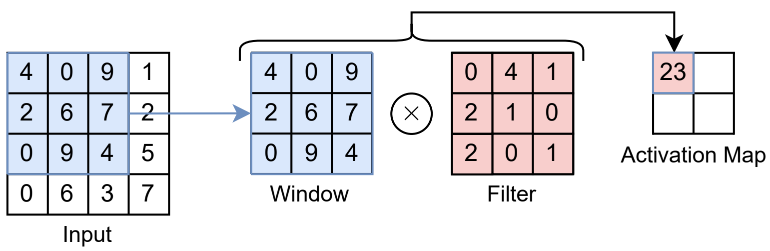


Figure 2.5: An example of the operation of a convolutional layer.

  - A **pooling layer** is a fixed function that acts as a downsampling operation. It reduces the number of parameters and computations in the network. It is typically inserted between two convolutional layers. Two common pooling functions are to compute the maximum or average for each patch of an activation map. Figure 2.6 shows an example of max and average (avg) pooling.

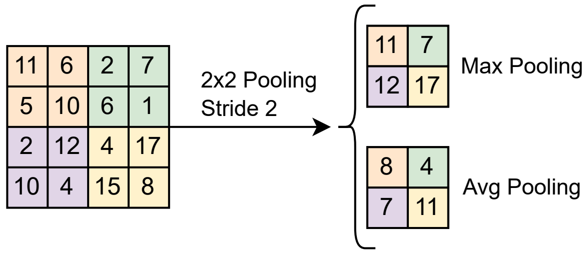


Figure 2.6: Examples of the max and avg pooling operations.

– Within a **fully connected layer**, all inputs of one layer are connected to each activation unit of the next layer. The input of a fully connected layer is the flattened activation map of a convolution or pooling layer. Flattening transforms a two-dimensional activation map into a vector. Fully connected layers are also used as the output layer of a CNN to perform the actual classification.

Over the years, several CNN architectures have been proposed for text classification, varying in the size, depth, frequency, and structure of the convolutional, pooling, and fully connected layers [190, 195, 255].

- **Recurrent Neural Network (RNN).** An RNN is an NN architecture for processing sequence data. Text is viewed as a sequence of words, enabling the capture of its structure and word dependencies. Each word (represented by a word embedding) is recurrently evaluated by incorporating information from the previously processed sequence of words. In an RNN, information propagates in loops from one layer to another. The output of neurons serves as their own input for subsequent computations.

  The following equations describe the inner workings of a cell:

$$h^t = \sigma(h^{t-1}, x^t; \omega) \tag{2.11}$$

$$o^t = \rho(h^t; \omega) \tag{2.12}$$

  In step t, the previous hidden internal state of the last step h^{t-1} and the current input x^t are fed into the network to compute h^t. σ and ρ are activation functions and ω is the weight matrix and bias of the network. The output o^t depends only on the hidden state h^t. An RNN has a strong ability to infer contextual meaning from the sequence of preceding words.

A Long Short-term Memory (LSTM) [152] is an evolution of the classical RNN described above. LSTMs can store long-term dependencies while traditional RNNs struggle to do so [32].

- **Encoder-based LLMs**. In addition to providing text encodings, encoder-based LLMs can also be used as end-to-end text classifiers. Figure 2.7 illustrates an LLM for text classification. The input text is first tokenized and passed to the LLM. For each input token, the LLM computes a transformer block. Devlin et al. [88] suggest passing the transformer block associated with the [CLS]-token, a contextualized vector representing the entire input text, to a single-layer FFNN to perform classification directly. In general, we can use any hidden state from any internal layer of BERT for classification. For example, Li et al. [225] suggest feeding the hidden states into a CNN, while Fang et al. [106] add an RNN on top of BERT to perform classification.

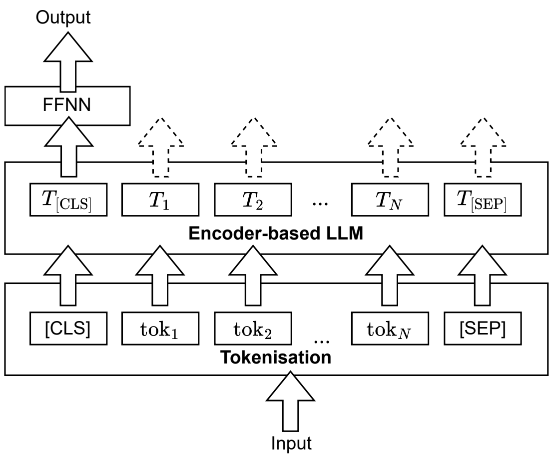


Figure 2.7: Simplified scheme for using an encoder-based LLM for text classification.

An NN is a highly flexible structure in which layers can be assembled in almost any compatible way. Over time, countless combinations and variations of FFNNs, CNNs, and RNNs have been proposed for text classification. Some classifiers also attempt to combine the advantages of CNNs and RNNs [208, 379]. Yu et al. [400] provide a general review of RNN-like architectures, while Gu et al. [127] present an overview of CNNs.

2.1.7 Classifier Validation

Text classifiers must be highly accurate (good enough) when deployed to meet stakeholder needs [183]. The question arises regarding how accurately a trained classifier solves the intended task. A validation step is needed to build trust and provide evidence of a classifier's effectiveness. Especially since classifiers will almost never be 100% accurate, there is always a risk of misclassification [87]. Since ML-based classifiers are very complex and the derived rules are difficult or impossible for a human to understand or interpret, the only way to validate the classifier is usually an empirical approach. The validation process should highlight the correctness of a classifier, i.e. how it is expected to perform during deployment. Several validation metrics have been established to assess the classification performance of ML models [136].

In the validation process, a classification model makes predictions $y_1, y_2, ..., y_N$ for each example $x \in X_{test}$ in a test dataset of size N. The predictions are then compared pairwise with the known ground truth values $\hat{y}_1, \hat{y}_2, ..., \hat{y}_N$, where y_i is the predicted class label of the ith sample and $\hat{y}_i$ the corresponding true value (ground truth). There are several established metrics for evaluating the sensitivity of application domains to different types of errors [345].

- **Accuracy.** The most basic performance metric is the accuracy. It describes the fraction of all correctly classified instances over all samples within a training dataset. This fraction is defined as:

$$\text{Acc} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\hat{y}_i = y_i) \tag{2.13}$$

  where $\mathbb{I}$ is an indicator function. However, the accuracy has a major drawback. It reports highly misleading values on imbalanced datasets. Consider a dataset where 99% of the samples do not correspond to the positive class. A classifier that always responds with the negative class would immediately reach an accuracy of 99%, since 99% of all classifications would be classified correctly.

- **F1 score.** The F1 score is another well-established metric for assessing the generalizability of a classifier. The F1 score is calculated per class, requiring some form of aggregation when there are more than two classes.

  In the **binary setting** (C = 2), the F1 score is estimated as the harmonic mean of the *precision* and *recall* of a classifier. For a class c the precision and recall distinguish between four types of classification outcomes. These are *true positives* $TP_c = |\{i|\hat{y}_i = c \wedge y_i = c\}|$, *true negatives* $TN_c = |\{i|\hat{y}_i \neq c \wedge y_i \neq c\}|$, *false positives* $FP_c = |\{i|\hat{y}_i \neq c \wedge y_i = c\}|$, and *false negatives* $FN_c = |\{i|\hat{y}_i = c \wedge y_i \neq c\}|$. While true positives (TP) and true negatives (TN) are correctly predicted, false positives (FP) and false negatives (FN) are incorrect predictions. Precision is also referred to as *true positive rate* (TPR) and recall as *positive predictive value* (PPV). The TPR (Eq. 2.14) indicates the proportion of true positives in relation to all assigned values of class c. The PPV (Eq. 2.15) measures the proportion of agreements in class c.

$$TPR_c = \frac{TP_c}{TP_c + FN_c} \quad (2.14) \qquad PPV_c = \frac{TP_c}{TP_c + FP_c} \quad (2.15)$$

The F1 score (Eq. 2.16) evaluates the ability of the models to accept instances from both classes. It is calculated as the harmonic mean of the TPR and PPV. It indicates how many instances were correctly classified without missing a significant number of instances, e.g. underrepresented classes.

$$F1_c = \frac{2 \cdot PPV_c \cdot TPR_c}{PPV_c + TRP_c} = \frac{2 \cdot TP_c}{2 \cdot TP_c + FP_c + FN_c} \tag{2.16}$$

We need some form of aggregation when computing the F1 score in **multi-class settings** (where $C > 2$). Micro-averaging (denoted by μ) aggregates the contribution of each class. Each sample is equally weighted:

$$TPR_\mu = \frac{\sum_{i=1}^{C} TP_i}{\sum_{j=1}^{C}(TP_j + FP_j)} \quad (2.17) \qquad PPV_\mu = \frac{\sum_{i=1}^{C} TP_i}{\sum_{j=1}^{C}(TP_j + FN_j)} \quad (2.18)$$

Macro-averaging (denoted by M) first calculates the metrics independently for each class. Then, the arithmetic mean of the resulting metric is calculated across all classes. Thus, macro-averaging gives equal weight to all classes:

$$TPR_M = \frac{\sum_{i=1}^{C} TPR_i}{C} \quad (2.19) \qquad PPV_M = \frac{\sum_{i=1}^{C} PPV_i}{C} \quad (2.20)$$

The macro or micro F1 score is calculated using the respective aggregated TPR and PPV within the Eq. 2.16. Macro-averaging is mainly determined by the majority class, since large classes have more influence. Micro-averaging is favored when class imbalance is expected and minority classes are most important. Since all classes are equally weighted, samples of minority classes become more important. Macro-averaging is preferred when the goal is to maximize the number of correct predictions made by the classifier and no class is more important than the others.

There is no single best metric for validating text classifiers. The most suitable metric depends largely on the objective and the costs associated with producing and correcting certain types of errors. Any validation metric raises the question of "*How good is good enough?*" [183]. While there is no clear consensus on this question, it depends on the specific requirements of the domain.

2.2 Prediction Uncertainty

ML models are inherently imperfect to some degree due to vagueness, lack of specificity, and conflicting knowledge about the task being solved. Noise and shifts in the data, as well as model misconceptions and limitations, cause inputs to be misinterpreted by the ML model, leading to incorrect classification results [148]. In general, classification models cannot reliably assess their trustworthiness [113]. The following section introduces the concept of estimating prediction uncertainty in text classification as a first step towards more applicable predictions.

2.2.1 Motivation

It is challenging to achieve error-free text classification using ML models [356]. Well-trained text classifiers typically do not achieve 100% accuracy even on the dataset they were trained on, and even worse on unseen data. Inputs are expected that the model cannot adequately interpret, resulting in incorrect and unreliable model behavior. Misclassifications are an inevitable consequence of various deficiencies and misconceptions that traditionally remain unnoticed. Typical weaknesses of ML-based classifiers are that the learned classification rule does not generalize well to shifting data [260], or that a model does not adapt to unseen data [312]. Both cases lead to less reliable and likely incorrect predictions. Nevertheless, practical applications require high classification performance and reliability for automated text classification to be effective and practical. Questions arise about how good a classifier is, whether individual predictions can be trusted, and how to communicate potential difficulties encountered.

The actual predictive quality of an ML-based classifier can only be evaluated empirically. As described in Section 2.1.7, the classification performance is validated using an unseen and typically sparse test dataset. Validation results are tuned by adapting and revising the model until it reaches a satisfactory level of classification performance that allows it to be used operationally. During operation, predictions are usually blindly accepted and assumed to be nearly as good as the test performance. However, the operational classification performance is likely to be much worse than in laboratory settings. This discrepancy is due to the simplified modeling or validation settings used during model development [117]. Additionally, the test and training datasets are usually a random sample of the same data distribution, while real-world data is unknown and subject to change [260].

Validation metrics primarily assess the average classification performance of a model based on a historical dataset. We can only assume that the model will perform similarly on data from the same distribution as the test data. The transferability of the model to unseen data instances remains unknown and can only be approximated. For example, consider a classifier with an accuracy of 90%. It is generally assumed that one out of ten predictions will be wrong. However, the test accuracy is only an approximation of the assumed classification performance. If 100 easily judgeable inputs are given to the model, no misclassifications are likely to occur. The exact amount of hard-to-evaluate text input is likely to produce a disproportionate number of misclassifications. The input itself strongly influences the reliability of an individual prediction, which cannot be accurately determined by averaging over the entire input space. Traditional accuracy-based validation metrics cannot be used to judge the reliability of individual classification results. They do not distinguish between a lucky guess and systematic evidence. We need other mechanisms to provide insight into the reliability of individual predictions.

Automation in high-risk domains such as autonomous driving [107] and medical image analysis [120] has highlighted the need for profound and high-fidelity model behavior, as incorrect predictions would have immense impacts on the well-being of people and their environment. Some even argue that emphasizing the reliability of individual predictions is essential for the safety of artificial decision-making [357]. Even in less critical areas such as text classification, highly accurate and reliable ML models are desirable because the occurrence and correction of misbehavior is still very costly.

Uncertainty estimation [113, 164, 220] is a tool for making explicit the uncertainty underlying individual predictions. Uncertainty deals with the ability of a model to recognize its limitations. Its quantification aims to estimate the ability of models to fail in recognizing the patterns of an input [356]. It is intended to provide valuable insights into artificial decision-making and ultimately help to make more informed decisions. Uncertainty is estimated to characterize the variability of predictions. Prediction uncertainties indicate the presence of noise or imperfect information, which is highly desirable to inform about unreliable predictions. However, there is no universally accepted definition of uncertainty because it is not a monolithic concept and takes various forms. It is generally

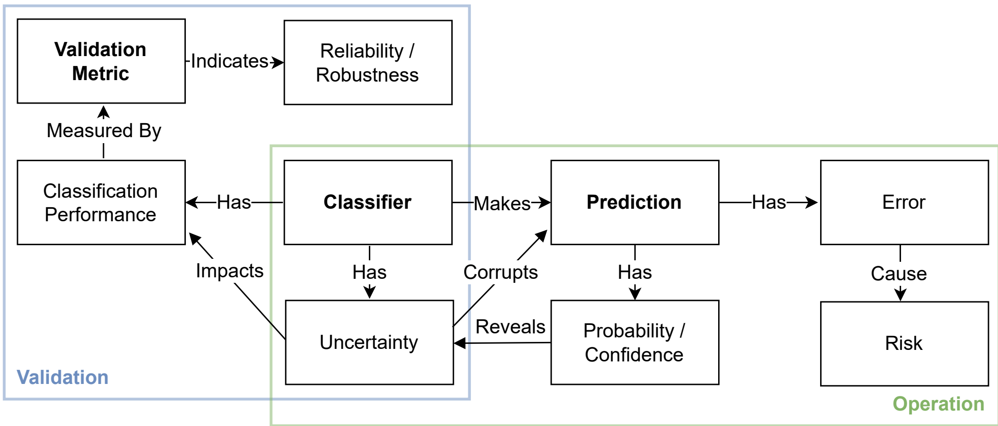


Figure 2.8: A metamodel illustrating different aspects describing the limitations of text classifiers.

hypothesized that highly uncertain predictions are highly unreliable and likely to be wrong [148] and that uncertainty information could help to manage the risk of ML-based classifiers [113, 356].

2.2.2 Overview

A perfect text classifier $\hat{f}$ which always predicts the correct label $\hat{y} = \hat{f}(x)$ for any $x \in X$ is practically unattainable due to inherent inaccuracies [87]. The exact relationship between X and Y remains unknown and can only be approximated. When describing potential limitations of classification models, the terms *classification performance*, *reliability*, *robustness probability*, *confidence*, *uncertainty*, *error* and, *risk* are commonly used to describe correlated but different measures of potential inaccuracy.

Classification Performance. The *classification performance* serves as a tangible measure of a model's reliability. This assessment is made during model validation using metrics such as the F1 score. Classification performance is also commonly referred to as *accuracy*, which should not be confused with the validation metric of the same name.

Reliability. Text classifiers should be as *reliable* as possible. Reliability is the degree to which predictions can be considered correct for a wide variety of typical inputs.

Robustness. *Robustness* is similar to *reliability*, but it describes the ability to perform well under varied and unexpected inputs [47], while reliability covers typical inputs. A perfectly reliable and robust classifier would cause no risk and be free of uncertainty.

Probability. A class *probability* measures the likelihood of an inferred class label and is directly reported by most classification models. A class probability represents the probability that an input belongs to the predicted class. The class with the highest probability is taken as the result of the classification.

Confidence. Class probabilities are also referred to as the *confidence* of a model, because they indicate the resilience of a classifier in its class outcomes. Since we are only dealing with text classifiers that model probability distributions over class labels, this thesis uses the terms *confidence score* and *class probability* synonymously.

Uncertainty. The concept of *uncertainty* is different from probability. Generally, uncertainty refers to a state of imperfect or unknown information. In ML, uncertainty characterizes the variability of prediction outcomes. Here, uncertainty is generally interpreted as a lack of confidence in a prediction [21, 334]. Prediction uncertainties can originate from various sources [87] and significantly influence the quality of predictions, for example by causing misclassifications. Risk in ML models arises especially in the presence of epistemic uncertainty (Section 2.2.4) [370].

Error. The classification *error* generally describes the difference between the actual classification outcome and its true value $\hat{y}_i$. The prediction error can be expressed directly as $\epsilon_i = \hat{y}_i - p(y = c|x_i, \omega)$, where ϵ depends on x_i and ω. The error ϵ arises from the effect of potential uncertainties such as inaccuracies, noise, and misspecifications. For instance, a classifier cannot reduce the aleatory uncertainty (Section 2.2.4), and some classification errors remain.

Risk. ML-based classification models are fraught with *risk*, as their classification performance is subject to uncertainty. In this context, risk can be outlined as the expected value of the cost of harm [370]. Consequently, risk encompasses both the cost associated with a classification failure and the likelihood of its occurrence. Uncertainty in model behavior introduces a potential for failure and thus contributes to risk. Formally, the risk of an ML model is quantified as the expected classification error that we aim to minimize during the training process [369].

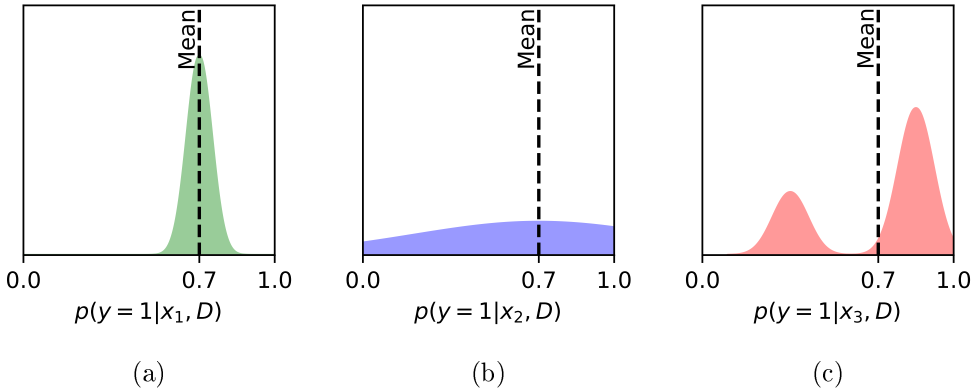


Figure 2.9: Difference between class probability and uncertainty. All predictions have the same mean probability ($\mu = 0.7$) but different levels of variability and dispersion (uncertainties).

Class probabilities and prediction uncertainties are two related but distinct concepts for assessing the reliability of individual predictions. Most text classifiers learn some probability distributions $p(y = c|x)$ that indicate the likelihood of label assignments. Since classifiers are deterministic by default, each prediction provides a point estimate of the underlying probability distribution (confidence score). However, the confidence of a classifier does not necessarily represent the uncertainty of the underlying probability distribution.

Figure 2.9 illustrates the approximated probability distributions given three text inputs $x_1, x_2, x_3 \in X$ with $x_1 \neq x_2 \neq x_3$. A vertical line highlights the mean class probability. As illustrated, all probability distributions have the same mean of 0.7. Thus, all three predictions were made with the same level of confidence. However, the degree of variability and dispersion of each prediction distribution is substantially different. In Figure 2.9a, the probability distribution of the prediction is tightly clustered around the mean class probability with low variance. Therefore, a point estimate is likely to accurately describe the true distribution, resulting in a relatively confident prediction with low uncertainty. In Figure 2.9b, the underlying probability distribution is similar to a uniform distribution. Thus, there are a variety of point estimates that could fit the distribution well. Here, the model is much more uncertain about which estimate to choose. Figure 2.9c shows a distribution that could be interpreted in two ways, both with fairly high probabilities. However, no clear decision can be made, resulting in a much more uncertain prediction. The predictions in Figures 2.9b and 2.9c are typically more uncertain than those similar to Figure 2.9a. Unfortunately, conventional classifiers only provide a point estimate without providing information about the underlying probability distribution.

We can easily determine confidence scores and classification errors. They are an integral part of the traditional classification pipeline, i.e., they are used during inference, training, and validation. Unfortunately, empirical evidence of the uncertainty of classification models is generally not available [113]. The question is why a point estimate $p(y = c|x, D)$ is not a good measure of uncertainty. Intuitively, a class probability of 0.99 seems more trustworthy and certain than a class probability of 0.76. The primary reason lies in the tendency of point estimates from deterministic models to be disproportionately high [129], often resulting in misleading interpretations. Deterministic softmax probabilities may poorly represent the true underlying probability distribution, deviating significantly from the distribution mean. For instance, a classifier may incorrectly assign very high class probabilities to inputs from regions that do not appear in the training dataset. In addition, typical NN models do not account for all types of uncertainty, limiting their usefulness as meaningful uncertainty indicators [113].

2.2.3 Sources of Uncertainty

Prediction uncertainties can arise from various sources throughout the traditional ML pipeline [320], significantly impacting the reliability of predictions. Common sources of uncertainty include misconceptions, bias, ambiguity, noise, and a lack of representativeness [87]. Listed below are the most common sources of uncertainty within text classification pipelines [87, 118, 164].

Pre-processing and Feature Misconceptions. Converting natural language text into a numerical representation poses uncertainties. While text cleaning aims to improve the quality of text instances, it is uncertain which cleaning and feature selection steps are optimal. In addition, natural language itself may be ambiguous, introducing uncertainty in its interpretation. The quality of text input varies widely, including its expressiveness, sonority, grammatical correctness, and noise level. Text cleaning steps can cause significant information to be lost or become noisy.

Model Misconceptions and Bias. There is no single classification model that is the best choice for every text classification task. All proposed models are empirical approximations and are wrong in some way. Any ML-based classification model introduces uncertainty due to its inherent limitations [87]. It is unknown which model is best suited to solve a given task, which parameterization is optimal, and how close it can approximate the function it is trying to learn. Training an ML model is a stochastic process that introduces statistical uncertainties in parameter selection, computational errors, numerical approximations, and truncations. Potential errors in the architecture specification of a classifier can also lead to uncertain predictions.

Training Data Misconceptions. Highly accurate models require a reasonable amount of high-quality training data. However, training data is typically sparse, and only a random sample of its true distribution is available. The problem of which and how many data instances to use for training introduces uncertainty. Furthermore, it is unknown whether there are still regions in the data that are sparsely represented.

Task Inherent Difficulties. The ambiguity, subjectivity, and fuzziness of natural language text are incompatible with the technical assumption of a seamless division of text into predefined classes. The boundaries between classes are not necessarily deterministic and may overlap, leading to uncertainty in the results.

Human Related Errors. Human decisions and activities play a critical role in the text classification process, contributing to uncertainties. Unintentional errors during modeling, design, construction, and operation can introduce uncertainties. Additionally, human annotators are vital in collecting training data by manually labeling examples, which introduces a degree of subjectivity. Human labeling errors, such as incorrect or inconsistent annotations, further contribute to uncertainty in the learning process.

2.2.4 Types of Uncertainty

Although there are many potential sources of uncertainty, it is common to categorize uncertainty as either epistemic or aleatory [87, 251]. Epistemic uncertainty is based on a lack of knowledge that can be reduced by providing more data, while aleatory uncertainty is considered irreducible and inherent in the data. Some authors, such as Malinin and Gales [245], also consider uncertainty arising from distributional changes relative to the training data (distributional uncertainty) as an additional type of uncertainty. Figure 2.10 illustrates the different types of uncertainty in classification using a simple linear model.

Epistemic Uncertainty. Epistemic uncertainty, also known as data uncertainty, arises from insufficient or missing knowledge. If a model is not sufficiently trained, it will have difficulty making reliable decisions. A sufficient amount

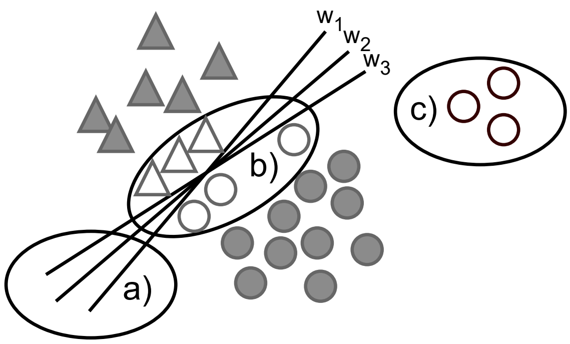


Figure 2.10: A linear classifier illustrating different types of uncertainty: a) epistemic uncertainty, b) aleatory uncertainty, and c) distributional uncertainty. Adapted from Mena et al. [251].

of training data is needed to determine appropriate model parameters during training. Figure 2.10 shows that when the training data is sparse, it is more complex and less certain for the classifier to generalize to new instances. This is because many possible model parameters can describe the underlying ground truth phenomenon, each with different errors. Therefore, epistemic uncertainty arises from the model weights and occurs because the model is unsure which weights best explain the data. High uncertainty arises in domains with little or no observed data for training. The ambiguity and uncertainty will decrease as more data is available in that area. With unlimited training data, a consistent learning algorithm will eliminate all epistemic uncertainty.

Common sources of epistemic uncertainty are labeling errors in the training data (human-related errors) and low data coverage, i.e., the training data does not represent the true distribution of a phenomenon (training data misconceptions). A second sign of epistemic uncertainty is the use of an ignorant and poorly fitting classification model (model misconception). Different models have different learning capabilities, which can lead to over- or under-fitting of the training data. For example, a linear classification model may not be able to fit more complex nonlinear relationships in the training data, resulting in highly uncertain predictions. Choosing a better classification algorithm on the same dataset can reduce the epistemic uncertainty.

Aleatory Uncertainty. Aleatory uncertainty is rooted in the nature of the phenomena and is beyond the control of a model. It is considered to be an inherent property of the data distribution, contaminated by noise and randomness. Therefore, aleatory uncertainty is not affected by the classification algorithm and cannot be reduced. A typical case of aleatory uncertainty occurs when the input is too ambiguous to make a clear decision (task-specific). In general, aleatory uncertainty exists when the best possible prediction can only be made with some degree of uncertainty. For example, suppose we attempt to predict the outcome of a coin toss. In this case, the best prediction for an unmanipulated coin can only be made with 50% certainty due to inherent randomness. Aleatory uncertainty can be further divided into *homocedastic* and *heteroscedastic* uncertainty. Homoscedastic uncertainty assumes that the natural noise is consistent across all inputs. Thus, homoscedastic uncertainty represents the average uncertainty of a given task. The heteroscedastic uncertainty is the uncertainty that depends on the input.

Distributional Uncertainty. It has been shown that unfamiliar inputs lead to much more unreliable and uncertain predictions [304]. So-called distributional uncertainty occurs when the instances to be predicted do not come from the distribution of the underlying training data [245]. In the case of distributional mismatch, the model is confronted with unfamiliar inputs for which it has not been sufficiently trained to make reliable decisions. Data shift occurs when actual real-world data changes with respect to the training data [260]. Shifted data is still a valid input, but it is not part of the actual task on which the model was trained. For example, a classifier trained to detect the sentiment of movie reviews will not work well on insurance fraud. Distributional uncertainty is a subset of epistemic uncertainty because it is rooted in a lack of knowledge. Thus, epistemic uncertainty is often used as a method for detecting out-of-distribution examples [148].

2.2.5 Uncertainty Estimation

Epistemic and aleatory uncertainty can be feasibly determined within NNs [113, 210]. Recent approaches to estimating prediction uncertainty are mainly based on probability theory. The focus is on so-called Bayesian Neural Networks (BNNs) [55, 121, 360], which are based on Bayesian statistics [43]. BNNs provide a framework for statistically correct modeling of epistemic and aleatory uncertainty in NNs. The following section briefly introduces BNNs; for a more technical introduction, see Jospin et al. [174].

For the introduction of BNNs, let us first recapitulate how conventional NNs work. Given a training dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$ of size N. The goal of a conventional NN is to learn an optimal parameterization ω that describes D in the best possible way. The resulting classifier f^{ω} is then used to predict the label of some x that was not used for training. The optimal weight is determined by minimizing a loss function via back-propagation, typically the mean square error (MSE). Since the optimal weight is a point estimate and has a true value, the resulting classifier f^{ω} is deterministic. That is, it will always behave identically given the same input.

Within a BNN, a probability distribution called *posterior* is placed over the real-valued weights of the model $\omega = \{W_i, b_i\}_{i=1}^{L} \in \Omega$. The posterior can be computed by applying Bayes' theorem [121], as shown in Eq. 2.21.

$$p(\omega|D) = \frac{p(D|\omega)p(\omega)}{p(D)} \tag{2.21}$$

Bayes' theorem is applied to estimate the posterior $p(\omega, D) = p(\omega)p(D|\omega)$, which indicates how well the weights ω describe the data D. The posterior allows for a statistically correct uncertainty quantification. The evaluation of the posterior requires the *likelihood* $p(D|\omega)$, the *prior* $p(\omega)$, and the *marginal likelihood* $p(D)$.

Likelihood. The likelihood of the data D given the weights ω, expressed as $p(D|\omega)$, describes how well the data D can be explained by the weights ω. It expresses the distribution of weights compatible with the data. We can assume that all samples $(x, y) \in D$ are independent and equally distributed (i.i.d.). Thus, the likelihood can be calculated as the product of all individual conditional class probabilities $p(y|x, D)$, which is usually assumed to be Gaussian, that is:

$$p(D|\omega) = \prod_{i=1}^{N} p(y_i = \hat{y}_i|x_i, \omega) \tag{2.22}$$

For classification, a softmax likelihood is taken as the conditional class probability. The softmax likelihood is the probability that an input x belongs to class c given a classifier f^{ω} with weights ω. The softmax likelihood is given by:

$$p(y = c|x, \omega) = \frac{\exp(f_c^{\omega}(x))}{\sum_{k=1}^{C} f_k^{\omega}(x)} \tag{2.23}$$

Prior. The prior $p(\omega)$ encodes the initial belief about how the weights ω are distributed. It describes what is known about the model weights before observing the training data. This initial belief about the prior must be set before training. Typically, BNNs assume a Gaussian prior. The prior is updated as the BNN is trained.

Marginal Likelihood. The evaluation of the marginal likelihood $p(D)$ requires solving an integral over the weight space $\mathbf{\Omega}$, that is:

$$p(D) = \int_{\Omega} p(D|\omega)p(\omega)\, d\omega \tag{2.24}$$

The weight space of an NN is highly dimensional, and averaging over all possible weights is a challenging task that cannot be solved analytically [39, 113].

Predictive Posterior. The predictive posterior distribution $p(y|x, D)$ is used for inference. It describes the probability that a single input x belongs to class c, given training data D. The predictive posterior, also called the Bayesian model average, is the prediction distribution weighted by the likelihood for all possible weights ω, given by:

$$p(y|x, D) = \int_{\Omega} \underbrace{p(y|x, \omega)}_{\text{Aleatory}}\, \underbrace{p(\omega|D)}_{\text{Epistemic}}\, d\omega \tag{2.25}$$

The predictive posterior captures the total uncertainty inherent in the prediction. Given a weight ω, $p(y|x, \omega)$ captures the aleatory uncertainty, and $p(\omega|D)$ represents the epistemic uncertainty. The predictive posterior is not based on a single weight estimate, but is marginalized over the entire posterior. In this way, all possible weights are taken into account. A deterministic model would only use a single fixed weight ω to compute a class probability $p(y = c|x, \omega)$ based on the softmax outcome, which only captures aleatory uncertainty.

Unfortunately, the predictive posterior (Eq. 2.25) is intractable and cannot be evaluated analytically because the marginal likelihood $p(D)$ requires averaging over all possible weights ω. The intractability arises from the high dimensionality of the weight space $\mathbf{\Omega}$, which makes integration over all possible weights impractical. Several approximation techniques have been proposed to exploit the advantages of Bayesian modeling while remaining applicable [39, 113]. A common approach is to approximate the untraceable posterior $p(\omega|D)$ with an easier-to-evaluate variational distribution $q(\omega|\theta)$. This variational distribution is estimated by finding the parameters θ of $q(\omega|\theta)$ that minimize the Kullback-Leibler (KL) divergence [204]. The KL divergence measures the similarity between two distributions. This replaces the intractable problem of averaging over all weights with an optimization problem defined as [121]:

$$\hat{\theta} = \operatorname*{argmin}_{\theta} \left\{ KL(q(\omega|\theta)||p(\omega|D)) := \int_{\Omega} q(\omega|\theta) \log \frac{q(\omega|\theta)}{p(\omega)p(D|\omega)}\, d\omega \right\} \tag{2.26}$$

The variational distribution $q(\omega|\theta) \approx p(\omega|D)$ replaces the posterior in Eq. 2.25 when evaluating the predictive posterior $p(y|x, D)$. Finding a tractable approximation to the true posterior density that minimizes the KL divergence is also known as *Variational Bayes*. However, Eq. 2.26 poses another intractable integral that cannot be computed in a closed form.

Since exact Bayesian inference over the weights is still intractable, several approximation techniques have been suggested [39, 113, 210]. Most techniques solve the marginalization problem by approximating the predictive posterior using a Monte Carlo integration as follows:

$$\begin{aligned} p(y|x, D) &= \int_{\Omega} p(y|x, \omega) p(\omega|D)\, d\omega \\ &\approx \int_{\Omega} p(y|x, \omega) q(\omega|D)\, d\omega \\ &= \mathbb{E}_{q(\omega|D)}[p(y|x, \omega)] \\ &\approx \frac{1}{T} \sum_{t=1}^{T} p(y|x, \omega_t) \end{aligned} \tag{2.27}$$

Common techniques are: *Bayes by Backpropagation* [39], *Monte Carlo Dropout* [113] and *Ensembles* [210].

Bayes by Backpropagation (BBB). Blundell et al. [39] propose BBB, a practical solution for learning probability distributions over the weights ω of an NN. BBB assumes that the weights have a Gaussian distribution. Thus, the network weights $\omega = (\mu, \sigma)$ consist of means μ and standard deviations σ to be learned. A variational distribution $q(\omega|\theta) \approx p(\omega|D)$ is defined to approximate the true posterior $p(\omega|D)$. The authors of BBB propose a gradient descent compatible cost function that approximates Eq. 2.26 to learn the optimal parameters $\hat{\theta}$. The cost function is given by:

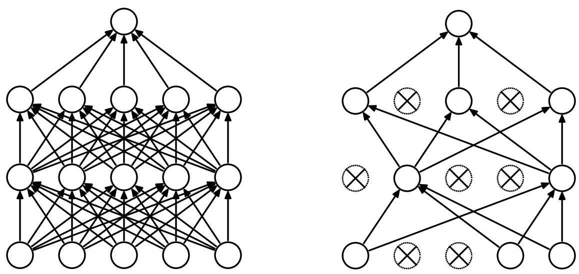
$$L(D, \theta) \approx \sum_{t=1}^{T} \log q(\omega_t|\theta) - \log p(\omega_t) - \log p(D|\omega_t) \tag{2.28}$$

where ω_t is the tth sample drawn from the variational posterior $q(\omega|\theta)$.

Monte Carlo Dropout (MCD). Gal and Ghahramani [113] show that the regularization technique *dropout* [349] can be interpreted as a Bayesian approximation of a Gaussian process. Dropout randomly disables or "drops" connections between neurons and previous layers during training. It is implemented by adapting the original feed-forward operation of a neuron by multiplying a vector $r = \text{Bernoulli}(p) \in \{0, 1\}$ of independent Bernoulli random variables with a layer input vector a, that is:

$$z = w(ra) + b = \sum_k w_k(r_k a_k) + b \tag{2.29}$$

where each element r_k has the probability p of being 1. Applying dropout to an NN results in training multiple "thinned" networks with numerous shared weights. Given a network with n neurons, dropout can be interpreted as jointly training n^2 possible weights $\omega \in \{\omega_i\}_{i=1}^{n^2}$ simultaneously within a single classifier. Figure 2.11 illustrates the dropout operation. Dropout was originally proposed as a regularization technique to prevent overfitting of ML models and to make them much more robust by achieving a lower generalization error [349].



(a) Standard NN. (b) Thinned NN.

Figure 2.11: A standard NN and a thinned NN after applying dropout [349].

Dropout variational inference is performed by training an NN with dropout enabled before each weight layer and additionally enabling dropout during inference. By randomly dropping units, the model result is a random variable and is no longer deterministic. Performing multiple probabilistic forward passes results in an approximation of the posterior, that is $q(\omega|\theta)$. Each dropout configuration corresponds to a different sample of weights $\omega_t \sim q(\omega|\theta)$. This corresponds to variational inference. The intractable integral of the predicted posterior is approximated by performing T Monte Carlo samplings (Eq. 2.27), each using a random sample of weights. Thus, MCD approximates the class probability $p(y|x, D)$ using T deterministic models $f^{\omega_1}, f^{\omega_2}, ..., f^{\omega_T}$ parameterized by T sampled weights $\omega_1, \omega_2, ..., \omega_T \sim q(\omega|\theta)$, where $q(\omega|\theta)$ is the dropout distribution.

Ensembles. The use of ensembles is another effective mechanism for approximating Bayesian marginalization (Eq. 2.25) [210]. An ensemble is a committee of multiple similar ML models, each applied to the same classification task. An ensemble is built in two steps. First, multiple classifiers are trained separately on data sampled from the same data distribution. Then, inference is performed on each model individually, and the resulting class probabilities from all models are averaged to form a single prediction. Due to the stochastic nature of training, each model learns a slightly different version of the classification function $f : X \to Y$. A boost in classification performance is expected as multiple models do not tend to make the same error [116, 210]. Building an ensemble is straightforward because it does not require any changes to the architecture or source code of the individual models. Only the results of multiple models need to be collected and averaged.

Ensembles are not a true Bayesian approach. According to Lakshminarayanan et al. [210], Bayesian inference can still be interpreted as marginalization over an infinitely large ensemble of models. They also show that averaging the output of an ensemble is an approximation to the intractable integral of the posterior probability (Eq. 2.25). Formally, an ensemble of size M can be described as a set of different weights $\{\omega_i\}_{i=1}^{M}$, each initializing a classifier $f^{\omega_1}, f^{\omega_2}, ..., f^{\omega_M}$. The weights $\omega_1, \omega_2, ..., \omega_M$ are randomly initialized and the corresponding models are trained independently. The posterior probability is computed by approximating the intractable integral by averaging the results of M independent models.

2.2.6 Uncertainty Quantification

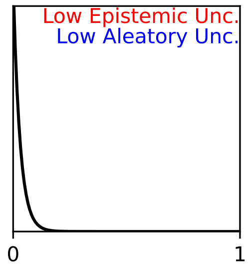
Modeling probability distributions over model outputs provides a natural way to investigate the uncertainty of a classifier. Uncertainty estimates can be derived by analyzing the statistical dispersion of the output distribution [39, 113, 207, 251]. Generally, uncertainty quantification techniques aim to assign high uncertainty values to highly unreliable model results and vice versa.

The *empirical variance* of a stochastic process is traditionally taken as an approximation of its uncertainty [113]. The higher the variance of a prediction, the more uncertain it is. In text classification, the variance of a class label y given an input x is given by:

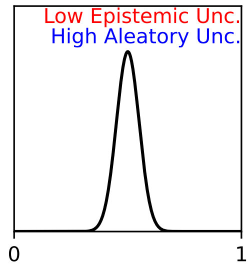
$$\begin{aligned}\sigma_c^2(x) &= \mathbb{E}_{q(\omega|D)}[p(y=c|x,\omega)^2] - \mathbb{E}_{q(\omega|D)}[p(y=c|x,\omega)]^2 \\ &\approx \frac{1}{T}\sum_{t=1}^{T} p(y=c|x,\omega_t)^2 - \Big(\frac{1}{T}\sum_{t=1}^{T} p(y=c|x,\omega_t)\Big)^2\end{aligned} \tag{2.30}$$

The mean-variance over all class outcomes can be used to capture the variance of all class outcomes given an input x, that is:

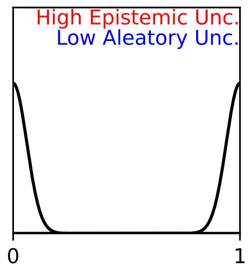
$$\sigma^2(x) = \frac{1}{C}\sum_{c=1}^{C}\sigma_c^2(x) \tag{2.31}$$



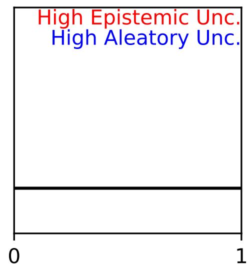
(a) High confidence with low variance.



(b) Mean near decision boundary and low variance.



(c) Conflicting outcomes are equally likely, each with high confidence and low variance.



(d) A uniform distributions represent a state of maximum uncertainty.

Figure 2.12: Different configurations of probability distributions, showcasing variations that lead to either low or high levels of epistemic and aleatory uncertainty.

However, the empirical variance does not account for all types of uncertainty and is rarely used in text classification. It captures the variability of the model (epistemic uncertainty), but does not explicitly capture the natural randomness and noise of the data (aleatory uncertainty). A prediction with zero variance can still be highly uncertain, e.g., by reporting a mean class probability of $\mu_c = 0.5$ in a binary classification task.

Both the variance and the mean of the posterior influence the uncertainty of a prediction. The epistemic uncertainty accounts for the variance of a model, which can be reduced or even eliminated by using more sufficient training data. The aleatory uncertainty is revealed directly by the class probability $p(y|x, \omega)$ (Eq. 2.25), given some weight ω. Values close to 1 or 0 contain less aleatory uncertainty, while values close to 0.5 contain high uncertainty in the sense of not being able to clearly distinguish between classes. The variance and the mean are not completely unrelated since the predictions have a small variance when the mean confidence is close to 0 or 1. Figure 2.12 shows that a prediction may involve high epistemic and no aleatory uncertainty, no epistemic and high aleatory uncertainty, both high epistemic and high aleatory uncertainty, or neither.

Especially for classification tasks, the *empirical mean* is also an important factor of prediction uncertainty [113], indicating a lack of confidence in the predictions. A number of confidence-based uncertainty measures have been proposed to quantify the uncertainty of predictions, including:

- **Least Confident** (**LC**) [76]. Measures the uncertainty as a lack of confidence in only the highest class probability:

$$LC[y|x, D] := 1 - \max_c p(y = c|x, D) \tag{2.32}$$

- **Smallest Margin** (**HM**) [325]. Measures the uncertainty as the difference between the most (c_1) and the second most (c_2) probable class outcomes:

$$HM[y|x, D] := \operatorname*{argmin}_x p(y = c_1|x, D) - p(y = c_2|x, D) \tag{2.33}$$

- **Shannon's Entropy** (**H**) [334]. Shannon's Entropy[1] is the classical measure of uncertainty. It quantifies the average information content of probabilistic data. Shannon's Entropy takes into account the lack of confidence in all class probabilities and is defined as:

$$H[y|x, D] := -\sum_c p(y = c|x, D) \log_2 p(y = c|x, D) \tag{2.34}$$

- **Ratio of Confidence** (**RC**) [325]. Describes the ratio between the highest and second highest class probability, that is:

$$RC[y|x, D] := \frac{p(y = c_2|x, D)}{p(y = c_1|x, D)} \tag{2.35}$$

All of these metrics are derived from $p(y = c|x, D)$ (Eq. 2.27), but can also be computed from a single prediction $p(y = c|x, \omega)$ given some weight ω. For example, Shannon's entropy can be rewritten as $H[y|x, \omega] := -\sum_c p(y = c|x, \omega) \log_2 p(y = c|x, \omega)$ using a single weight ω. Eqs. 2.32, 2.33 and 2.35 can be adapted analogously. In the latter case, no epistemic uncertainty is captured.

[1] Shannon's entropy is generally defined as: $\mathbb{H}(p(x)) := -\sum_x p(x) \log_2 p(x)$, given some probability $p(x)$.

However, a single forward pass can be interpreted as a poor approximation of the mean of a prediction probability distribution.

When modeling probability distributions, more robust estimates of the mean can be made. Also, additional information hidden in the distribution can be considered in the uncertainty assessment. When using Bayesian modeling, the following established uncertainty measures can be applied:

- **Variational Ratio** (**VR**) [110]. The VR ratio is a vote-based uncertainty measure that evaluates uncertainty as the dispersion of the nominal class outcomes. It refers to the number of predictions that do not agree with respect to the mode class. The number of predictions that occur in the mode class is denoted as f_{mode} using T forward passes:

$$VR[y|x, D] := 1 - \frac{f_{mode}}{T} \tag{2.36}$$

- **Mutual Information** (**I**) [114, 158]. Measures the uncertainty of an input x according to the mutual information between the predictions and the model posterior, defined as:

$$\begin{aligned} I[y|x, D] &:= H[y|x, D] - \mathbb{E}_{q(\omega|D)}[H[y|x, \omega]] \\ &\approx H[y|x, D] + \frac{1}{T}\sum_{c=1}^{C}\sum_{t=1}^{T} p(y = c|x, \omega_t) \log_2 p(y = c|x, \omega_t) \end{aligned} \tag{2.37}$$

According to Eq. 2.37, an ML model is highly uncertain about predictions when there is high marginal uncertainty (high $H[y|x, D]$) and disagreement about predictions with high certainty (low $\mathbb{E}_{q(\omega|D)}[\mathbb{H}[y|x, \omega]]$). Mutual information is commonly referred to as BALD (*Bayesian Active Learning by Disagreement*), which denotes an uncertainty-based Active Learning query strategy [158].

- **Uncertainty by Kwon** (**KW**). Kwon et al. [207] propose a method for quantifying uncertainty that exploits the relationship between the variance and the mean of predictions. Their approach allows the decomposition of prediction uncertainty into its aleatory and epistemic parts. KW does not require changing the network architecture to capture both types of uncertainty. The uncertainty measure is given by:

$$KW[y|x,D] \approx \underbrace{\frac{1}{T}\sum_{t=1}^{T}(\mathrm{diag}(p_i)) - p_i p_i^{\intercal}}_{\text{Aleatory}} + \underbrace{\frac{1}{T}\sum_{i=1}^{T}(p_i - \bar{p})(p_i - \bar{p})^{\intercal}}_{\text{Epistemic}} \quad (2.38)$$

where $\bar{p} = \frac{1}{T}\sum_{i=1}^{T} p_i$ and $p_i = p(y|x, \omega_i)$ is the softmax probability.

There are a number of other approaches to quantifying uncertainty in predictions that are not described in detail here. For further approaches, we refer to the surveys of uncertainty quantification methods provided by Gawlikowski et al. [118].

2.3 Conclusion

This chapter has introduced the task of text classification. It has presented a generic pipeline for building an ML-based text classifier, including data cleaning, feature engineering, model selection, and classifier validation. For each step, established implementations have been outlined. The focus was mainly on deep learning approaches using NNs.

Each subtask in the ML pipeline can introduce uncertainty into the classification results, affecting the reliability and variability of the predictions. Bayesian statistics provides a framework to naturally model uncertainty in NNs. Two main types of uncertainty have been identified. First, predictions are affected by aleatory uncertainty, which describes irreducible noise and randomness in the data being predicted. Second, epistemic uncertainty characterizes the lack of knowledge to make reliable predictions, which can be addressed by providing more training data. Several approaches for approximating BNNs and quantifying uncertainties have been discussed. Being aware of prediction uncertainties is a step towards more transparent and responsible predictions. However, simply estimating the uncertainty of classifiers does not make them more reliable. It is necessary to take measures that allow effective integration and use of uncertainty information in the decision-making process.

Chapter 3

Applicability Challenges of ML-based Text Classifiers

Publication. This chapter is based on the paper "Why do we need Domain-experts for End-to-end Text Classification? – An Overview" [9] in 2023. My contribution to this paper encompasses the entire research process, including the literature review, synthesis, and presentation of the results, as well as leading the writing process.

Contribution. User comments are a valuable source of information with high economic value for many real-world use-cases. However, ML-based text classification faces several challenges that limit its applicability. This chapter first outlines real-world use-cases for text classification that we focus on in this thesis. These include the classification of app reviews, problem reports, and social media posts. Second, it explores and describes vital challenges that hinder the applicability of text classifiers. We provide an overview of current challenges and group them into four main types: data-centric, model-centric, human-centric, and applicability-centric challenges.

3.1 Examples of Real-world Use-cases

In online communication, the rise of social media, online forums, and other digital platforms has enabled individuals to express themselves or communicate with others through text messaging. Many domains, including online marketplaces (such as *Amazon*[1] or *App Store*[2]) or news media (such as *The Washington Post*[3]), have recognized the potential of allowing their audiences to provide explicit user feedback [234, 239, 283]. With the increasing popularity of online communication, these domains share the need to manage large volumes of user

[1] https://www.amazon.com/
[2] https://www.apple.com/app-store/
[3] https://www.washingtonpost.com/

comments. User feedback holds great promise for facilitating their services, enabling them to better understand their audience, make data-driven decisions, and achieve their business goals. However, enabling user comments also includes obligations, such as regulatory compliance.

Text classification models and techniques provide a generic and domain-independent solution space for text classification systems. This thesis focuses on better exploiting the potential of explicit user feedback by enabling more applicable text classification. To evaluate the proposed HiL frameworks, we consider several real-world use-cases from the domains of **software engineering**, **online journalism**, and **social media analytics**.

3.1.1 Feedback and Review Classification

App stores provide a platform for downloading software and also allow users to provide valuable feedback about it. Users who have downloaded an app can typically rate it on a scale of 1 to 5 and write a review message. App stores have become an important communication channel between app users and developers [7]. Using and making sense of app store feedback for developers and analysts is of great interest [139]. For instance, app stores have become the top prioritized channel for bug reports [7]. Developers can easily access the users' view and use this feedback to drive the development of their applications [239].

Feedback from app stores is a key component of **data-driven software engineering**. Data and analytics serve as the primary drivers for making informed decisions throughout the software development process, such as utilizing user feedback to identify, prioritize, and manage the requirements for a software product [241]. Particularly promising are reviews from app stores [7, 139, 239], which commonly contain bug reports, feature requests, questions, or valuable user experiences. Feedback and review classification have become critical factors for developers to stay competitive in a rapidly evolving app market. As the number of reviews increases, so does the need for applicable text classification.

3.1.2 Issues Ticket Classification

The continuous management and tracking of software defects, tasks, and other work items is critical for effective and successful software maintenance and evolution [178]. For this reason, software projects use Issue Tracking Systems (ITS) such as Jira[4] to efficiently collect, document, and track issues in software systems that need to be resolved. An issue is a unit of information that typically includes a title, description, and several properties such as type, status, priority,

[4]https://www.atlassian.com/software/jira

and links to other issues [259]. Users can submit different types of issues, such as asking questions, suggesting features, or reporting bugs with varying levels of quality [178, 259].

Software developers are required to timely react and **resolve issues reported in ITS**. However, for popular software systems, tens or hundreds of issues are reported daily [178], and manually labeling issues can be labor-intensive, error-prone, and time-consuming [104]. Assigning labels (issue types) to issues is critical for prioritizing, handling, and resolving issues [178, 179], i.e., filtering relevant issues for assignment to appropriate teams and channels. For example, ML-based systems are highly desirable to accurately determine the types of issues to effectively handle and prioritize them [179].

3.1.3 Hate Speech and Offensive Language Detection

Hate speech and offensive language can take many forms, ranging from overt racism, sexism, and discrimination to subtle microaggressions, trolling, and cyberbullying. In addition to harming individuals and communities, these forms of expression can have far-reaching societal consequences. They can damage the reputation of service providers and alienate users. To combat the negative consequences of such content, the development of ML-based hate speech and offensive language detection systems has received considerable attention in the field of ML [80, 256, 267, 296, 383]. Especially in online journalism, hate speech and offensive language have become a major obstacle [20, 285, 297].

Online journalism is the digital dissemination of news and information through Internet-based platforms. It involves the creation, curation, and presentation of news content through various digital media such as websites, social media, and mobile applications. Online forums have become an integral part of online journalism [234, 307]. They allow users to share personal opinions, valuable feedback and corrections on journalistic content such as articles or videos. While the potential of leveraging user feedback within online forms is widely recognized [234], the quality of user feedback varies widely. Freedom of expression has led to a dark side of online discourse [297]: the spread of hate speech and offensive language, which is a major threat to news organizations [40, 285]. Approaches are needed to reliably identify hate speech and offensive language in public discussions. Developing reliable content moderation systems to block unwanted content remains an open research challenge [283].

3.1.4 Sentiment Analysis

Social media provides a way for users to communicate and network with each other. In addition to posting images and videos, users interact by exchanging text messages and comments. Social media has become an essential space for monitoring public opinion. According to Kaplan and Haenlein [180], *Social Media* is defined as "*a group of Internet-based applications that build on the ideological and technological foundations of Web 2.0, and that allow the creation and exchange of user generated content*". Social media data is vast, noisy, distributed, unstructured, and dynamic, making it difficult to process. The field of **social media analytics** refers to the collection and monitoring of social media data in order to extract and make sense of it to support (business) decisions.

One of the most common use-cases for social media analytics is sentiment analysis [105, 225, 243, 266]. Sentiment analysis aims to classify a text as expressing a positive, neutral, or negative affect, mood, or emotional tone (sentiment) [249]. More granular levels of sentiment have been explored, such as a $[-5, +5]$ range [132]. Sentiment analysis is commonly used to understand the sentiment regarding certain entities such as movies [243], online courses [225], or software features [132]. Among other aspects, sentiment analysis allows practitioners to understand customer satisfaction, identify areas for improvement, pinpoint specific problems, or highlight customer preferences.

3.1.5 Topic Classification

User comments are typically associated with distinct topics, such as themes, aspects, or features within a discussion. Topic classification is concerned with structuring extensive collections of text into recurring topics. It can be applied to classify incoming user requests into relevant topics such as politics, sports, technology, and entertainment to assist editors in receiving content of interest. In addition, topic classification is commonly used to understand which aspect of a product is most frequently discussed or to recommend relevant text based on user interests [212]. Topic classification can also identify emerging trends or popular topics in news articles or support tickets across different classes over time, helping journalists or support members focus their coverage on areas of high interest.

Dividing text into topics helps organize, understand, and filter large amounts of data. The frequency of topics reveals what users are talking about. Unlike *topic modeling* [2], topic classification is based on supervised rather than unsupervised ML. Therefore, the topics must be known in advance.

3.2 Classification of Applicability Challenges in ML-based Text Classification

Text classification research continues to propose ML models, architectures, and best practices that outperform previous benchmarks. However, there remains a gap between research settings and the requirements and expectations of real-world use-cases, which negatively affects their applicability. This gap affects state-of-the-art approaches and all ML-based text classifiers in general. While the operation of ML systems requires proper requirements engineering activities and techniques [242], there are still data, model, human and applicability-centric challenges that negatively impact the delivery and acceptance of ML-based systems.

While accuracy metrics drive most research, many practical considerations are overlooked. Even the most accurate ML models may not immediately meet the requirements of the application domain. Common ML obstacles include a lack of labeled examples, limited hold-out accuracy, lack of user trust, runtime constraints, low data quality, natural fuzziness, and wrong expectations. The following section reviews and outlines critical challenges to the applicability of text classifiers, especially in real-world applications.

3.2.1 Related Surveys

Several literature surveys indicate different challenges to the applicability of text classifiers or ML models in general. For example, Brodley and Smyth [51] describe various problem-specific factors that influence the development process of ML models, including application, data, and human factors. Rahman et al. [300] report on a general survey of 80 practitioners outlining common challenges in ML application development, covering the entire ML pipeline. These range from data collection to model deployment and maintenance.

A significant source of challenges is the trend toward increasing amounts of data. Liu [231] highlights the issue of real-world datasets consistently growing in size, creating challenges for contemporary data analysis. This poses significant difficulties from a data management perspective [65, 380]. Wang et al. [380] examine the technical challenges of deep learning from a database perspective, including operation scheduling, memory management, and distributed training. Najafabadi et al. [265] outline challenges associated with deep learning in the context of big data. According to them, challenges arise from unstructured and heterogeneous data formats, noisy and poor quality data, fast-moving streaming data, high data dimensionality, algorithm scalability, imbalanced data distribution, and limited labeled data. A similar review is proposed by Zhou et al. [405]. They discuss key challenges and research directions for ML related to the use of big data, human labels, domain knowledge, and infrastructure. As computational power is critical to the success of ML methods, Cheng et al. [68] highlight the challenge that high storage and computational costs hinder the deployment of ML systems in situations where computational resources are limited.

Challenges also arise from quality aspects of how an ML system should behave. Vogelsang and Borg [376] outline the quantification of quality targets such as explainability, freedom from discrimination, legal and regulatory requirements, and data requirements as the main challenges and objectives of requirements engineering for ML. Habibullah and Horkoff [133] investigate the applicability of deep learning from a requirements engineering perspective. They propose 25 non-functional requirements for applying ML models in production environments, including accuracy, reliability, explainability, fairness, and others. Maalej et al. [242] review challenges to the applicability of ML-based systems from a requirements engineering perspective. They focus on non-functional requirements and highlight the difficulty of defining measurable success criteria for quality attributes, such as the acceptable failure rate or risk. To ensure the deployment of AI (*Artificial Intelligence*) systems, Maalej et al. [242] outline aspects that need special attention to better meet user, system, and societal requirements. Khomh et al. [191] outline critical challenges to ensure the quality of ML systems from a software engineering perspective. These relate to the imperfect nature of ML models, their testability, and the gap between software engineering and ML activities.

Other surveys focus on the challenges posed by real-world use-cases. Adnan and Akbar [3] identify the lack of usability, where non-experts find it challenging to construct classifiers without great effort, and the need for substantial computational resources to train large models on large datasets as the main barriers to the widespread adoption of deep learning techniques. Rudin and Wagstaff [317] outline open challenges and research directions for the application of ML systems in real-world settings. Their research agenda focuses on ML-related challenges that affect society. Ittoo et al. [167] review challenges related to the application of text classification in real-world industrial environments. These challenges include dealing with heterogeneous data sources, short and informal text instances, a shortage of labeled data, and a lack of classification performance. Borrellas and Unceta [44] describe the economic implications of the problems of interpretability, fairness, and security of ML systems that are barriers to the development and mass adoption of ML for real-world use-cases.

Jöhnk et al. [173] review organizational readiness factors for AI systems: data accessibility, financial budget, AI readiness and infrastructure, openness to integration with black-boxes, and other necessary considerations. For example, they note that a lack of understanding and fear of AI can hinder the adoption of ML-based systems. Ashmore et al. [24] focus on the security of ML systems to ensure their intended use with respect to data management, model learning, model verification, and model deployment tasks.

Furthermore, D'Amour et al. [78] identify the challenge of *underspecification* that degrades the credibility of ML systems in applications. Underspecification causes ML systems to exhibit unexpectedly poor classification performance when used in real-world domains. They describe underspecification as a structural design flaw that can occur when classifiers are tested on hold-out data that follows the same distribution as the training data. A trained classifier is underspecified when many different predictors f can be trained from D that satisfy the test criteria but have systematically different generalization behavior. They show that even small changes can force an underspecified classifier to make different predictions in application domains [78].

3.2.2 Data-centric Challenges

Data-centric challenges to the applicability of ML-based text classifiers originate in the underlying data used to train them. These include inadequate and missing knowledge, as well as poor data quality.

Sparse, Inadequate or Incomplete Training Data. A lack of training data is a major bottleneck in text classification. Sparse, inadequate, or incomplete training data limits the ability of classification models to make accurate predictions. The inference is limited to identifying patterns familiar to the model from the training data. While text data is widely accessible nowadays, annotated data needed to train classifiers is still scarce [332].

Another major challenge for text classifiers is the generalizability of the training data. Generalizability refers to the ability of a trained ML model to effectively adapt to a range of inputs. The training, validation, and test datasets are generally assumed to represent the true text distribution $p(X)$. However, laboratory environments often differ significantly from real-world conditions. Since $p(X)$ is unknown, there is no indication of how well the training data represents the true data distribution that will be encountered during deployment. Blind-spots and unwanted biases may unintentionally manifest themselves in the dataset and distort the results [250].

Moreover, the data collected for training is usually historical and may reflect outdated and only static snapshots [201]. However, real-world data is highly dynamic and subject to change, much like the context of discussions and language, which evolve over time [95]. Common drifts to be expected lie in the distribution of the incoming data $p(X)$, the prior of the classes $p(Y)$, the conditional class probabilities $p(Y|X)$ such as changes in the decision bounds, and the conditional probability $p(X|Y)$ [115]. Undetected drifts negatively affects the reliability of classification results due to a less representative training data set. Furthermore, raw real-world data is much noisier, unfiltered, and of varying quality. Given these limitations, classifiers are not expected to perform as well in real-world settings as they do under controlled laboratory conditions.

Compromised Data Quality. Data quality is a relevant challenge for text classifiers because it significantly impacts the classification performance [75, 201]. The effectiveness of supervised ML models depends not only on the quantity but also on the quality of the data on which they are trained. Since data in the context of text classification consists of text and labels, we distinguish between label quality and text quality.

First, the quality of label assignment (*label quality*) measures the accuracy and consistency with which provided labels match the actual true labeling [336]. Label quality typically depends on the accuracy, consistency, and objectivity of the humans providing the labeled data (Section 3.2.4). If the labels are inaccurate or inconsistent, the classification performance will be adversely affected [300]. In addition, incorrect labeling in the test data can make it difficult to determine whether the model is making errors and to diagnose and fix problems. High label quality is critical to the success of text classifiers.

Second, the quality of the input text (*text quality*) is also crucial. High text quality is essential because classification is based on patterns and relationships in the data. Models cannot adequately interpret low-quality data, resulting in poor classification performance. In this case, the classifier may fail to learn properly, leading to inaccurate classification outcomes. The quality of generic data is traditionally measured along various dimensions such as accuracy, relevance, representation, completeness, accessibility, timeliness, and others [378]. Data quality is standardized in the ISO/IEC 25012 standard [166], which outlines 15 data quality characteristics, including inherent data quality aspects and system-dependent data quality. However, critics argue that such measurements are subjective [386] and that it is difficult to define success criteria for quality attributes [242]. The key dimensions of data quality for textual data are:

Accuracy. Accuracy refers to the extent to which a text aligns with its intended class. This classification ability is influenced by how accurately the features reflect the text presented and how accurately the text itself expresses its intended meaning. Highly accurate text instances are also free of typos, noise, and grammatical errors that interfere with the classification [4].

Relevance. Relevance refers to the degree to which a text fulfills the needs of the classification objective. A text instance is considered relevant if it aligns with the type of text the classifier is expected to encounter in real-world applications. For example, a highly relevant text for a class would contain numerous key terms and concepts associated with that class. Conversely, non-relevant data often consists of outliers or out-of-distribution instances that do not align with any predefined class [140].

Completeness. A complete text instance contains enough information to make accurate predictions. Concise, uninformative texts without context lack completeness and may not be related to any class. In terms of training data, a complete dataset would cover all words and contexts that occur in its application domain (high generalizability). Complete training data must cover a range of classes or categories and various text styles and formats. A typical indicator of incompleteness is the appearance of blind spots [26].

Consistency. Measures how consistent and free of errors a dataset is. This includes consistent use of language, i.e., the same key terms, concepts, contexts, and appearances. A high consistency makes it much easier for classifiers to assign labels accurately. Inconsistent text instances may contain words that are not expected for their labeling. Different text formats, structures, and domain-specific languages introduce inconsistencies that can negatively affect a classifier's classification performance.

Timeliness. Timeliness refers to the currency of the data. It describes the degree to which the data is up-to-date and represents the data a classifier is expected to encounter in its real-world application. In text classification, this means that the training reflects the current state of the domain. Outdated data may not reflect current patterns, resulting in poorly trained classifiers. Timeliness is especially important in text classification because natural language is constantly evolving, as is its context [5]. The classification performance of a classifier is known to degrade over time if the training data is not up-to-date.

In the context of ML, data quality is usually outlined as the degree to which a dataset serves a given task [128]. Data quality has a significant impact on the quality of predictions [387]. A high overall data quality is desirable because it provides a good approximation (high classification performance) of the true dependency of inputs and outputs. Learning powerful classifiers does not require arbitrary data, but good quality data [221].

3.2.3 Model-centric Challenges

Model-centric challenges to the applicability of text classification refer to characteristics of the classification model that negatively impact its application. These challenges are caused by the model itself, but can also be caused by missing, conflicting, or poor quality data (data-centric challenges). Typical model-centric challenges include a lack of performance and reliability, a lack of user trust, high computational cost, and model latency.

Lack of Classification Performance and Reliability. While recently proposed ML models for text classification continue to push the state-of-the-art classification performance [88], the performance still needs to be much higher for practical application. Even with high-quality training data, the most advanced text classifiers rarely achieve a perfect classification performance. Typically, the classification performance converges to a maximum reachable level that the model cannot exceed. Benchmarks indicate that classification performance well below 100% is expected for well-scoped tasks. [88, 144, 397].

In particular, even state-of-the-art text classifiers may fail to provide top classification performance. Often a classification performance of around 85% is reached [217, 277, 302, 304, 315]. In contrast, in the manufacture of goods and the provision of services, a minimum accuracy of 99.9997% (Six-Sigma [206]) is usually aimed at in order to be highly efficient and to ensure high process quality [358]. In the field of ML-based text classification, such nearly error-free process flows are not to be expected according to the current state of the art.

A concept closely related to classification performance is **reliability**. It is the ability of a model to perform well and consistently on a wide range of inputs [363]. Tran et al. [363] outline three dimensions of reliability for AI systems. To be reliable, a model must accurately represent its own uncertainty, consistently perform well in new scenarios, and efficiently adapt to new data. Reliability therefore presupposes a high level of classification performance in a broader context than the training and test data. Overall, a lack of classification performance and reliability leads to misclassifications.

Lack of Transparency and User Trust. While some traditional classifiers, such as DTs, are interpretable by design (as long as the decision rules are easy to interpret) [38]. Most recent classifiers, especially deep learning approaches, remain opaque and are by default in-transparent [59]. Such classifiers are black-boxes as they do not provide comprehensible insight into their inner workings [59]. It remains unclear why and how ML models derive at the given labels. Without human-readable explanations, practitioners can hardly be convinced by the classification.

The problem of worrying about trusting ML models lies in their fallibility and inability to explain their decisions and actions [23]. Trust describes the confidence of humans in artificial decision-making and is a primary reason for the acceptance and application of ML models [339]. Trust is necessary to create the willingness to rely on artificial decisions and deploy them in practice [309]. The key to building human trust is to understand and know the strengths and weaknesses of an ML model [56]. Typically, human users tend to trust models when they can understand why certain predictions were made. In general, black-box predictions should not be blindly trusted, especially when it has a large impact on its environment.

High Computational Complexity and Resource Consumption. Current research in text classification focuses mainly on classification performance rather than efficiency [353], which is not a problem at first because highly effective models are needed. However, ML models generally face a trade-off between classification performance and utilizing computational resources [294]. The trend is towards increasingly complex models that continue to outperform previous approaches [353]. As more powerful computing infrastructures become available, more complex classification algorithms are being developed, pushing the boundaries of the-state-of-the-art. Infrastructure and computational resources have become critical factors in keeping pace with current achievements and not falling behind.

State-of-the-art ML models for text classification require significant computational resources [88, 144, 397], making them challenging to deploy in lightweight environments such as edge devices. Computational constraints have become a significant bottleneck in model training and deployment. This can be challenging when high-performance hardware is unavailable, or memory constraints are an issue. Real-world product environments are typically not as powerful as the laboratory environments of research-intensive companies and universities. This prevents many practitioners from using state-of-the-art ML models to automate text classification.

Recently, the importance of alternative evaluation criteria has increased. The awareness of a sustainable and green IT landscape is gradually making its way into the field of ML [6, 205, 353]. User awareness is shifting towards sustainable and computational-aware use of artificial intelligence. Training deep learning models is particularly resource-intensive, requiring millions of operations to compute the values of the weights. The question is whether it is really necessary to use highly complex systems to extract the last percentage points of classification performance, or whether computational-aware ML models are sufficient.

Fuzzy Classification Objective. Natural language text is inherently fuzzy, and its interpretation is highly subjective. The complexity of a text may not match the uniqueness of classification goals, which strictly aim to separate classes. In contrast, class boundaries of real-world data are usually fluid and may overlap [394]. An observation may erroneously belong to none, several, or all classes, or an input x may belong to two or more classes with a non-zero probability. While classification models assign inputs to exact classes, natural language is somewhat vague with a certain degree of interpretation. For example, studies show that even domain experts disagree on the detection of hate speech [25, 383]. It is generally assumed that if domain experts cannot agree on a certain class membership, an ML model will not be able to do better [41]. The ambiguity of assigned class labels is also indicated by the common use of inter-annotator agreements, which are applied to resolve inconsistencies between multiple annotators [383].

A challenge of ML-based classifiers is that they may not be able to cope with the complexity of the problem to be solved. Intermediate possibilities between classes cannot be modeled, although they are present in the data. For example, sentiment labels are usually "positive" or "negative" [243]. However, text is often multifaceted and ambiguous. A text may contain both positive and negative sentiments. The true sentiment covers many more intermediate steps than are actually modeled. Therefore, some consider "neutral" to be a third category, so that labelers do not always have to choose between the two extremes. Others try to approach this diversity by considering more fine-grained classes. For example, the sentiment of product reviews can be classified according to more intermediate levels, such as "strongly positive" and "weakly positive" [105]. However, this leads to increased complexity and a more difficult labeling task, while still facing the problems of ambiguous boundary cases.

3.2.4 Human-centric Challenges

Human-centric challenges revolve around the role of humans in the text classification pipeline. In particular, this thesis focuses on data labeling, since text classifiers rely heavily on a large volume of high-quality but manually labeled data instances. This section describes the challenges posed by the resilience of human labelers and outlines the main obstacles associated with manual labeling.

Cost of Human Effort. Human labor is typically constrained by its high cost and tight budgets. Additionally, manual text classification is highly labor-intensive and time-consuming, requiring careful reading and reasoning for each text. For instance, Gray et al. [124] report that labeling 211 legal opinions engaged seven hired law students for over two months at a cost of several thousand dollars. Unfortunately, text classification requires large amounts of labeled training data. A tight budget for labeling usually limits the classification performance and reliability of classifiers. Saving human resources is critical to enabling the applicability of ML models to real-world domains. Reducing the cost of human involvement while maintaining certain quality constraints is a common goal of ML systems [332]. Human cost-effectiveness should be taken into account when designing ML systems. Although some researchers [125] consider a labeling budget of 1,000 instances as low-budget, such amounts would still require enormous manual effort.

Latency of Tooling. Another significant consideration in the applicability of text classification-based systems is the latency of the embedded ML model. Latency describes the delay between a model request and its response. Thus, latency is the time it takes for a classifier to perform training, or to receive an input, process it, and return a predicted class label. Typically, the latency of an ML model is correlated with its computational complexity [157].

Current state-of-the-art ML models raise concerns about their high latency, not only on edge devices [64], but also on exceptional infrastructure. Interacting with and refining ML models during their development can easily be a lengthy process, violating the time constraints of interactive systems [233]. In most cases, it is insufficient for humans to wait hours for an ML model to re-train before continuing to provide feedback [333]. The trade-offs between classification performance, computational complexity, and time constraints are often not adequately considered in the race for effectiveness. Most research papers do not even mention the training time and computational resources required [353].

Subjectivity, Biases and Misconceptions. During data labeling, humans must make decisions based on imperfect, complex, and noisy information. However, humans are subject to specific errors in their reasoning [306]. Sometimes humans make mistakes in labeling text, adding noise to the model [333]. Human judgment can be corrupted by several human distortions that negatively affect the quality of human feedback, including cognitive, perceptual, and motivational biases [295]. It may not always be possible to reach a global and objective consensus.

There are several reasons why human annotators may mislabel text instances. One factor is a lack of understanding and meaning, or missing context information. Mislabeling can also occur when annotators are unfamiliar with related concepts or terminologies. Such humans are typically referred to as non-domain experts. A typical example are crowd workers [374]. Studies show that a high variance in feedback quality between experts and non-experts is to be expected [344].

Labeling is also known to be a repetitive, tiring, and tedious task, leading human labelers to “turn off their brains” [8]. In such cases, the quality of provided labels may vary over time [109]. Also, when human labelers are under time pressure or need more time to read and think about a text carefully, they may mislabel it. In this case, labelers may not recognize the true essence of the text, leading to hasty and ill-considered decisions. Additionally, humans may misclassify text due to unintentional errors, such as misreading or selecting the wrong classification label.

Security and Misuse. Human feedback in the form of labels is fed directly into the model and directly affects its behavior. ML models are highly vulnerable to malicious user intent. So-called poisoning attacks [359] can be executed, where a subset of the training data is intentionally altered in some direction to change the behavior of the model during operation. Effectively preventing or detecting a poisoning attack is an open question [359].

3.2.5 Applicability-centric Challenges

Finally, this thesis considers challenges related to the broader applicability and use of text classifiers in real-world settings. The focus is on ensuring that ML solutions can be effectively implemented and sustained in practical scenarios. While many model- and data-centric challenges relate to quality issues, the challenge remains to specify acceptable yet feasible levels of these qualities [242].

Unrealistic Expectations. A common challenge in applying ML models is the presence of overly optimistic or unrealistic expectations about their capabilities and results [242]. Stakeholders may expect ML models or systems to solve complex problems perfectly or instantly, leading to disappointment when faced with the inherent limitations or uncertainties of these models.

For example, consider a team working on an ML system to automatically classify user feedback in an app store. Stakeholders, including app developers, may be excited about the potential of such ML systems to aggregate user feedback. They may expect the ML system to perfectly classify each user comment into its most relevant class, eliminating any misclassifications or inconsistencies. However, accurately classifying apps involves dealing with different nuances of review quality, app functionality, varying descriptions, and multiple classes that could apply to each review. In reality, ML systems often struggle to understand and reliably classify the nuances of each app, especially when reviews are complex, noisy, or ambiguous.

Managing and aligning stakeholder expectations with the actual capabilities and potential limitations of ML systems is critical to a successful deployment. It is important to educate stakeholders about the capabilities and limitations of ML models, emphasizing that these systems are probabilistic and may not always provide perfect solutions. Managing expectations by communicating realistic outcomes, potential risks, and uncertainties helps align stakeholders with the true capabilities of the technology [242].

Acceptable Levels of Quality Requirements. Quality requirements such as high classification performance, transparency, and misclassification tolerance are essential in ML projects. However, these requirements are often fuzzy, difficult to specify, and difficult to measure [376]. A significant issue lies in the undefined acceptance levels for these quality requirements [242]. For example, a generic requirement for an ML model might be "accuracy should be as high as possible". While suitable for research purposes, this vague specification is inadequate for responsible ML systems. Accuracy requirements must be explicitly defined to provide clear guidance to stakeholders and ML engineers. Without clear agreements and measurable levels, quality goals may receive minimal attention until serious acceptance issues arise. This underscores the need for a structured approach to defining and accepting quality requirements in ML projects.

Quality Assurance. Creating and maintaining value from ML systems over time requires ensuring the quality, reliability, and consistency of ML models in production. *Quality assurance* is a significant challenge that includes testing the classification performance of ML models across different datasets, monitoring for data drifts, assessing robustness against adversarial attacks, and maintaining the consistency of model outputs over time.

ML quality assurance ensures that ML models perform reliably, accurately, and consistently across different scenarios. This involves ensuring that the data used for training and testing is accurate, representative, and free of bias or error. Testing ML models against edge cases, anomalies, or adversarial attacks helps assess their resilience and robustness. In addition, to maintain accuracy and relevance over time, it is necessary to continuously monitor classification performance in production, detect drift, and frequently re-train models. Implementing robust quality assurance practices instills confidence in ML systems, ensuring they perform as intended and minimizing unexpected failures.

Tradeoff-analysis. Developing and tuning ML-based systems is challenging because not all quality requirements can be maximized due to inherent conflicts, competing objectives, or constraints [242]. Real-world applications require the consideration of various trade-offs.

For example, the pursuit of higher classification performance may conflict with sustainability goals, given the significant energy consumption and carbon emissions associated with tasks such as handling massive datasets and fine-tuning ML models. However, more complex models may provide higher classification performance but be less interpretable, requiring a trade-off between understanding the model's decisions and its predictive power.

Ensuring fairness in predictions can also impact the overall classification performance, requiring careful trade-offs to balance fairness and classification performance. Performing careful trade-off analysis helps to make informed decisions during the development and deployment of ML systems. It ensures that stakeholders understand the trade-offs involved and guides the prioritization of requirements [242]. A well-established trade-off analysis can help achieve a balanced and thoughtful approach to AI development, taking into account both technical and ethical dimensions.

3.3 Conclusion

There is a high demand for automating text classification tasks in real-world domains. However, the capabilities of ML-based text classifiers are limited, and such models lack applicability. In general, challenges to the applicability of ML models can be divided into data-centric, model-centric, human-centric, and applicability-centric aspects.

ML-based classifiers must be trained on a large and diverse dataset that contains a wide range of examples from each class. This allows the classifier to learn the key characteristics and features associated with each class and to accurately classify new pieces of text based on these characteristics and features. However, labeled training examples are limited and expensive to create manually. In addition, raw natural language text is typically of low quality, i.e., imbalanced, noisy, inaccurate, and inconsistent. Classifiers, however, require high-quality data to be trained accurately. Moreover, text is inherently fuzzy in its interpretation, and instances within the data are usually not separable into distinct classes. Unclear borderline cases are to be expected, which the classifier must still assign to a single class.

In addition, the choice and configuration of ML models pose a threat to their applicability. Classification models typically lack in classification performance and reliability. Misclassifications are to be expected. In the worst case, an ML model may not achieve the desired level of classification performance, making it ineffective in production. Decisions made by an ML model are also usually not transparent and lack user trust. Users generally do not take advantage of recommendations or suggestions that they do not trust. Furthermore, ML models are becoming increasingly complex, leading to a mismatch between research and deployable systems. Productive environments and laboratories often have different processing capabilities and requirements. In particular, real-time applications require low latency, which is typically not provided by state-of-the-art text classifiers.

Furthermore, the development of text classifiers requires human assistance, especially in the labeling of data. However, humans are unreliable, and some degree of mislabeling is to be expected due to unintentional errors, bias, or subjectivity. It is also challenging to manage the right expectations about what ML models can provide and their limitations in order to maintain user acceptance and highly applicable ML solutions. In addition, defining acceptable levels of quality requirements that may be correlated is challenging. As it is unrealistic to address all challenges simultaneously, i.e. due to data availability, data quality or computational constraints, a first step towards more accepted ML is to perform a trade-off analysis to help stakeholders and applicants become aware of potential limitations [242]. Summarizing the relevant findings of this chapter:

- ML-based text classification is a promising and dynamically evolving field of research. There is a strong desire to use automated text classification to uncover hidden insights or to filter certain types of content in numerous use-cases.

- ML-based text classification remains a challenging task with numerous applicability challenges. Purely automated solutions are typically corrupted by uncertainties, misconceptions, and model inherent limitations. Some degree of misclassification is to be expected.

- Research to address critical challenges in text classification is sorely lacking. For instance, even the most advanced text classifiers have shortcomings, typically achieving a classification performance of around 80%. As a result, their applicability in real-world environments remains questionable, often necessitating manual classification.

- Most research in text classification has focused on accuracy metrics to evaluate the effectiveness of classifiers. However, other critical factors such as computational cost, latency, or risk assessment are often neglected or not considered.

- Current text classification research is mainly focused on maximizing automation and taking humans out of the ML loop. There is a lack of research investigating the cost-benefit ratio of interactive ML approaches.

Part II

Solution

Chapter 4

Human-in-the-Loop Machine Learning

Publication. This chapter builds on the paper "Why do we need Domain-experts for End-to-end Text Classification? – An Overview" [9] in 2023. As the sole author of this paper, I contributed to all aspects of the research, development, and writing process. In addition, parts of this chapter are based on the paper "Towards Visual Data Science - An Exploration" [365] in 2020. My contribution to this paper concerns literature acquisition, synthesis, and writing. Parts of the chapter are also derived from the 2024 paper "Design Patterns for Machine Learning-Based Systems With Humans in the Loop" [10], where my contributions involved reviewing, extracting, and structuring best practices from the existing literature into cohesive design patterns. I also played a lead role in writing the paper.

Contribution. This chapter outlines the emerging Human-in-the-Loop (HiL) approach - a novel direction and computational paradigm for ML. HiL involves integrating human feedback into the ML process. Human feedback is seen as an integral part of facilitating problem-solving. HiL offers a promising solution to many of the limitations and challenges of purely automated text classifiers and ML in general, as detailed in the previous chapter. This chapter reviews previous implementations, perspectives, and definitions of HiL, and outlines its scope, key applications, and enabling technologies. In the current literature, we have found that the research area of HiL is rather convoluted and that there is a lack of engineering knowledge on how to effectively design HiL systems. Based on a literature review, we provide a catalog of seven training and five operation patterns to guide software engineers in selecting and implementing appropriate HiL solutions.

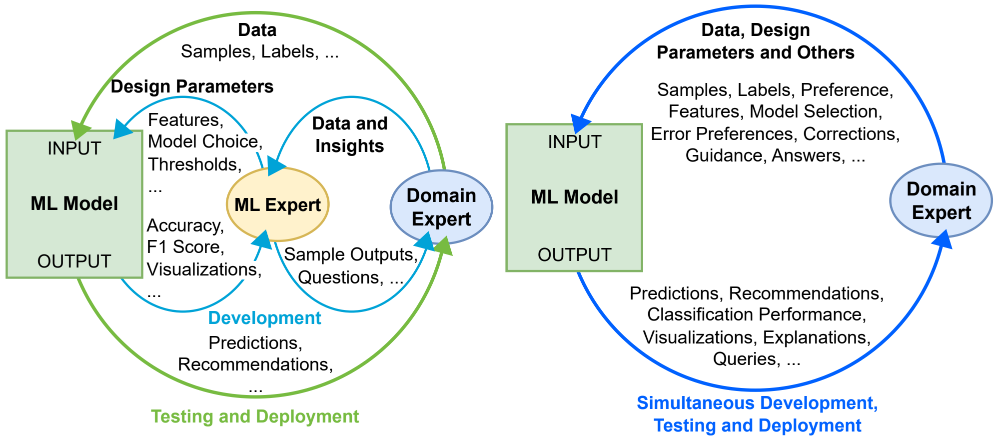
4.1 Motivation

Semi-automated ML approaches have gained prominence to overcome and mitigate the challenges that limit the applicability of text classifiers during development and deployment [8, 155, 184]. The need for semi-automated ML approaches arose from the observation that some ML models require analytical judgment, while others can be significantly improved or accelerated by letting humans interact with them [8, 103, 155]. According to Sacha et al. [319] the full potential of ML models cannot be realized without human participation.

A core characteristic of semi-automated ML approaches is the continuous support of ML models by human feedback. Human background knowledge, abilities, and expertise are closely intertwined with the capabilities of ML models. Allowing humans to interact with ML models aims to overcome critical challenges of purely artificial processing [194] and ultimately increase the applicability of ML approaches in real-world settings. The idea is simple: humans can provide additional information or make course corrections to ML models to achieve better processing behavior.

The concept of interactively adapting and supporting ML models with human feedback is commonly referred to as the **Human-in-the-Loop** (HiL) [155] approach. Its research area covers a variety of semi-automated ML approaches that are characterized by being supported by human feedback before, after, or during the learning phase. The humans is considered an essential part of the overall ML system to maintain its functionality and usefulness. However, humans have contributed to refining and creating ML models since its inception in the late 1970s. Humans have always been essential in various stages of the traditional ML pipeline [8]. Individual steps such as selecting and labeling data, engineering feature representations, configuring algorithms, evaluating models, and identifying ways to improve the overall classification performance require manual execution. However, there are significant differences between how humans are involved in the traditional ML approach and HiL.

Figure 4.1 compares the two approaches. In the traditional ML approach, the development process is handled solely by ML experts who have technical knowledge but no domain knowledge. Typically, ML models are created through a trial-and-error process, requiring a significant amount of time and human resources [145]. The lengthy, asynchronous, and technical nature of the traditional ML pipeline limits the ability of domain experts to influence the final models, even though they possess valuable domain knowledge and expertise. The involvement of domain experts is limited to providing data, answering domain-specific questions, negotiating with ML experts, or providing feedback



(a) Traditional ML paradigm (b) Human-in-the-Loop

Figure 4.1: Comparison between the traditional ML paradigm and HiL. Adapted from Amershi et al. [8].

on the learned model [8]. To address this issue, the HiL approach aims to empower domain experts to directly adapt ML-based systems for their own needs and purposes [8]. According to Karmaker et al. [181] a domain expert is *"a person who is fluent in the domain where ML is being applied but has minimal knowledge of how ML itself works"*. As shown in Figure 4.1, HiL eliminates the need for ML experts and enables domain experts to adapt machine processing in a much faster and more direct way. The value of HiL lies in its ability to incorporate human expertise and knowledge into the analysis that machines cannot provide on their own.

Another emerging approach aiming to overcome the strong dependency on ML experts is end-to-end learning. It aims to make it easier for non-ML experts to use ML models and to speed up the development and deployment process. End-to-end classifiers rely on neural models that do not require the explicit modeling of individual ML steps, such as pre-processing and feature engineering. ML capabilities are made available to individuals who are not ML experts, while increasing the effectiveness of the learning process [181]. AutoML [145] goes a step further by offering complete automation of the application of ML models to real-world problems with software that automates the entire ML pipeline. Eliminating manual steps in model training and development increases efficiency. Only the formulation of the problem and the provision of data remain in the scope of humans. AutoML aims to eliminate the need for ML experts. Although AutoML aims to make efficient and applicable ML models available to everyone, pure automation of the ML pipeline does not address the critical shortcomings of the classifiers themselves [215]. Human supervision is still required, as the ML model is not necessarily better than a manually designed and configured model. The increased applicability of AutoML comes mainly from automating tasks that would otherwise have to be performed by ML experts, or from preventing human errors during the design process.

The AutoML approach may at first appear to contrast with HiL, as it aims to take humans out of the ML loop instead of leveraging their knowledge. However, both approaches share the goal of making ML systems more applicable to real-world domains. Auto ML aims to provide an easy-to-deploy optimal end-to-end learning approach, while HiL aims to overcome the applicability challenges of ML models in practice. For text classification, both approaches can be combined seamlessly. For example, an AutoML pipeline can be used to build the initial optimal ML model, and then HiL can be applied to interactively adapt, refine, and improve it.

4.2 Scope

Several HiL solutions for ML have been proposed, covering a wide range of use cases and application domains [69, 182, 329, 388]. The following section outlines this heterogeneous landscape of existing interpretations of HiL. It defines what HiL covers and constitutes in the context of ML.

HiL encompasses a variety of closely related and overlapping concepts characterized by the collaboration of humans with ML models. In the current literature, the scope of HiL comprises many different terms, including Interactive ML [103], Human-centered ML [61, 347], mixed-initiative ML [82], Hybrid Intelligence [84], Human-AI Teaming [29], or Human-AI Partnership [273], and lies in the intersection of ML [257] and Human-Computer-Interaction (HCI) [91]. HiL aims to improve problem-solving compared to purely automated or purely manual approaches. Typical goals of HiL are more accurate [229], trustworthy [366], and fair [403] problem-solving or faster model development [332].

In recent years, a small number of HiL-related surveys and taxonomies have emerged, attempting to capture the scope of the HiL approach. However, the current literature lacks a unified perspective on what HiL actually covers. Wu et al. [392] outline HiL data processing methods for ML. They identified three main HiL categories: data pre-processing (interactive parameter tuning), data annotation, and iterative labeling (finding the essential samples to label driven by user experience). Mosqueira-Rey et al. [262] propose a HiL survey that focuses on how an ML model can learn from human feedback. They identify three

HiL approaches, which are using humans to label data (same as data annotation [392]) and having humans provide information in a more focused, frequent, and incremental way (same as interactive labeling [392]). They further depict *Machine Teaching* as a HiL approach, where humans control the knowledge they want to transfer to the ML model. In the taxonomy of design knowledge for combining the complementary strengths of humans and ML models by Dellermann et al. [84], human assistance is limited to feature engineering, parameter tuning, and training. Jiang et al. [171] suggest a taxonomy of interactive ML that states that recent work on interactive classification deals with interactive labeling, interactive feature engineering, and parameter space analysis. All of these major surveys and taxonomies outline the scope of HiL as being limited to the learning phase of a model, which ends when the model is deployed. However, a focus on training leaves many limitations of text classifiers untouched.

In contrast, Dudley and Kristensson [98] outline that human activities in HiL span the entire ML pipeline, including feature selection, model selection, and data labeling, as well as activities such as quality assessment and deployment. Similarly, Wang et al. [381] consider the evaluation and deployment steps of the ML pipeline as part of the HiL approach. This allows for further data annotation, corrections, and model re-training. In the same sense, Amershi et al. [8] argue that allowing humans to critique model outcomes is essential for interactive ML approaches.

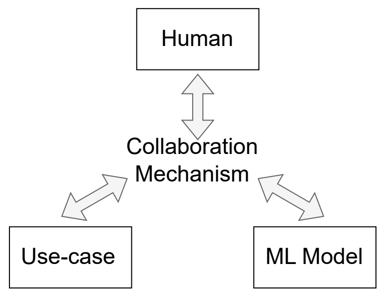


Figure 4.2: Key elements of a HiL system [98].

According to Dudley and Kristensson [98] HiL systems typically consist of four main components, as depicted in Figure 4.2. These components are: a human user, a model, data, and a user interface with collaboration mechanisms. The user can control the ML model by providing feedback or guidance to the model. The model is at the center of the process and receives its instructions from the data, which is a combination of existing and newly provided training examples. The interface connects the ML model and the user, offering interactive features and facilitating teamwork to accomplish a particular task. Collaboration mechanisms (Section 4.4) are used to facilitate human interaction with models. Potentially, humans can be involved in every step of the traditional ML pipeline, including data labeling, model selection and training, validation, and deployment [381].

4.3 Definition

At the time of the thesis, there is no generally accepted definition of HiL in the context of ML. Many attempts have been made, covering different aspects and use-cases of human-machine collaboration. The phrase *interactive Machine Learning* (iML) was originally introduced by Ware et al. [382] in 2001 to denote a collaborative approach that allows humans to develop classifiers. Users are empowered to construct Decision Trees based on their domain expertise. Previously, this task was accomplished only through automated means. Fails and Olsen Jr [103] were the first to use the phrase iML to describe a continuous *train-feedback-correct* loop for interactive model training. In their framework, humans continuously provide additional training data to a model until an acceptable level of classification performance is reached. This approach is commonly referred to as Active Learning (Section 4.5.3). Due to their data-centric focus, Fails and Olsen Jr [103] are often considered the forefathers of the contemporary interpretation of iML.

Porter et al. [290] further refine the iML workflow. They outline the methodology behind iML as an interactive process where humans and ML models work together on the same task at any stage of the ML pipeline. Additionally, they differentiate between two cases of human involvement. The human can be positioned at the end of the ML pipeline to validate the model outcomes or at the beginning of the pipeline to carry out the initial annotation task. Amershi et al. [8] contend that iML should adopt an extended user-centric perspective, prioritizing human factors and the quick, iterative nature of interaction cycles. They highlight the concurrent development and employment of ML models as a crucial component of iML. Additionally, they portray iML as a chance for domain experts and experienced practitioners to incorporate their knowledge directly into ML models. ML experts should not be required to modify and adjust the learning system to the needs of end users such as domain experts. Interactions in iML are much more focused, frequent, and incremental than those in traditional ML, resulting in more seamless and faster interaction cycles. Boukhelifa et al. [46] state that this user-centric nature of iML systems can play an essential role in mitigating the black-box effect of ML approaches.

One of the first to establish the phrase HiL is Holzinger [155]. The author uses the term HiL to describe "*algorithms that can interact with both computational agents and human agents and can optimize their learning behavior through these interactions*". This definition is at first surprising, since it allows machine agents instead of humans to be in the loop. However, the literature presented so far does not support a machine-only HiL perspective. HiL requires human agents in the ML loop. This definition focuses mainly on the machine-centric aspects of HiL, where models actively request feedback to support their behavior during the learning phase. In general, human-machine cooperation can be either user- or machine-centric. In a machine-centric approach, a model directly asks a human for information [332]. In a human-centric approach, humans select information and make it available to classifiers [33]. Dudley and Kristensson [98] outline the latter view. They define iML as "*a co-adaptive process, driven by the user, but inherently dynamic in nature as the model and user evolve together during training*". According to them, the best way to implement iML is to grant domain experts the power to develop their own learned concepts by collecting or labeling training data depending on their own requirements [98]. Their definition focuses on the user and illuminates the process of knowledge generation that occurs during the progressive interaction between humans and the model. Humans gain insight and knowledge about their data by observing its structure and model results, while machines learn from human feedback [321]. The acquired knowledge can help to further improve the quality of subsequent feedback. Endert et al. [101] go a step further and advocate a "Human-*is*-the-Loop" methodology in exploratory settings to emphasize the importance of seamlessly integrating human skills into the knowledge discovery process. Wondimu et al. [391] attempt to distinguish between iML and HiL. They outline iML as *an active machine learning technique in which models are designed and implemented with human-in-the-loop manner.* From their perspective, iML limits human involvement in model building. Whereas HiL describes the cyclical nature of the influence between a model and humans.

However, the application of HiL is not limited to model development. HiL can also be applied in deployment to further refine a model while it is being used in the field. Dellermann et al. [84] define the concept of *Hybrid Intelligence* as the "*the ability to accomplish complex goals by combining human and artificial intelligence to collectively achieve superior results than either could have done in separation and continuously improve by learning from each other*". Furthermore, they outline that both humans and machines can augment each other's knowledge, enabling superior problem-solving. In this sense, Wiethof and Bittner [389] differentiate between a HiL and *Computer-in-the-Loop* (CiL) methodology. CiL focuses on supporting humans by improving their work efficiency, while HiL is seen as an approach that encourages ML through human involvement and interaction with the ML model or learning system itself. Thus, they see HiL as a process in which a human augments machine intelligence and CiL as a process where a machine augments human intelligence. However, HiL does not necessarily aim for superior results. Monarch [258] defines the goals of HiL as: increasing the overall performance of a model, reaching the target performance faster, maximizing performance by combining human and machine intelligence or assisting human tasks to increase efficiency. Wang et al. [381] provide another definition, where HiL is described as a framework "*where model developers continuously integrate human feedback into different steps of the model deployment workflow.*"

Within this thesis, we define HiL as the following:

> Definition
>
> HiL describes a generic **semi-automated computational paradigm** in which **ML models** and **humans** interact, adapt, or learn from each other to improve the **applicability** of ML systems.

This definition captures the essence of the HiL approach on which we rely in the rest of this thesis. Applicability refers to the alignment of ML systems with the critical requirements of their use-case, such as a top classification performance and cost-awareness of human and computational resources. Furthermore, this definition emphasizes that humans and ML models are an integral part of an ML system. The scope of HiL is not limited to the training process, but includes all other steps in the ML pipeline, including model testing and deployment.

4.4 Collaboration Mechanisms

HiL aims at fast, efficient, continuous, and beneficial interactions between ML models and humans. While artificial decision-making is considerably cheap and fast, human involvement is usually the bottleneck of interactive approaches. To keep human involvement efficient and sparse, it is desirable to obtain high-quality feedback while avoiding redundant and unnecessary interactions. HiL systems must be designed so that humans can comprehend model outcomes effectively and be aware of when and how much they can trust them. In order to focus human attention on aspects where their assistance is needed, human involvement must be guided and supported. As mentioned earlier, a user interface is the central component of a HiL system, responsible for the bidirectional feedback between the user and the model [98]. The following section discusses three general user interface collaboration mechanisms that enable and support the fast exchange of high-quality feedback. These are **visualizations**, **explanations** and **uncertainties**.

4.4.1 Information Visualization

A user-centric design of HiL systems relies on the ability of human users to make effective use of their capabilities. This requires a certain level of insight into the data they are confronted with and an understanding of the behavior and outcomes of ML models. However, humans are easily overwhelmed by large amounts of data. This problem is often referred to as *information overload* and compared to finding a needle in a haystack. Roetzel [313] defines the information overload problem as "*a state in which a decision maker faces a set of information [...] that inhibit the decision maker's ability to optimally determine the best possible decision*". Typical causes include limited human resources to deal with the sheer volume of information, lack of time, and absence of appropriate tools. While classifiers are good at automating search and filtering tasks, they do not make it easy to understand and make sense of large collections of predictions.

A common approach to face the information overload problem is the use of visualizations. The field of *Information Visualization* deals with the computer-aided creation of graphical representations of data or concepts to amplify human cognition [58]. It is a way of presenting data in a meaningful way using charts, graphs, and other types of visualizations. The basic idea is that humans can effectively gain insight, make sense, and draw conclusions through a visual presentation [186]. Visualization can be used for a variety of purposes, including explanation, interpretation, communication, analysis, and decision-making. By harnessing the power of visual perception, visual representations of problem statements enhance human's ability to quickly and easily identify patterns and trends, find outliers and clusters, examine structures, observe relationships, and understand meaningful insights. These may not be immediately apparent and understandable in raw data. Card [58] suggests six ways in which visualizations can amplify human cognition, which are:

- Bringing increased resources to the human in the form of perceptual processing and expanded working memory.
- Reducing the search for information.
- Enhancing the detection and recognition of patterns.

- Enabling perceptual inference operations.
- Using perceptual attention mechanisms for monitoring.
- Encoding information in a manipulable medium.

While traditional visualizations are static, interaction has become an essential aspect of modern information visualization [337]. Interactivity enhances the ability to discover, explore, and understand large amounts of data, increases awareness of its meaning, and provides even greater insight and understanding. The most basic need for interactivity arises when the data does not fit into a static visualization. Interactions include, but are not limited to, filtering, zooming, brushing, and hovering [337]. Interactions make visualizations more dynamic and flexible. For example, users can focus on specific areas or scale the depiction of information. In particular, interactive visualizations are essential for HiL systems because they allow domain experts to interact with the ML model without requiring any additional programming [103].

4.4.2 Explanations

The general lack of transparency is a major challenge for the applicability of text classifiers (Section 3.2). Without insight into the model's reasoning process, it is unclear whether a classifier will perform adequately during deployment. Potentially erroneous or undesirable behavior remains hidden. To understand how a classifier works, it is necessary to provide human-interpretable approximations of the reasons behind the classification results. Of particular interest is the question of why a specific class was preferred over the others [35].

Explainability is a key enabler for HiL systems to be deployed in real-world use-cases [366]. The field of *Explainable Artificial Intelligence* (XAI) [79] deals with ML systems that attempt to explicitly extract human-understandable explanations of their predictions. An explanation is some kind of information that helps to verify a classification outcome. Explanations aim to open the black-box of classifiers and ensure that humans can understand and justify why certain classification results were delivered. They should enable humans to discover what a model has learned and how it can be further improved [79]. In addition, explanations aim to increase confidence in decision-making and provide a better understanding of the model's limitations, such as weak spots and biases. In general, explanations can improve a classifier's robustness and user trust, knowledge transferability, informativeness, accessibility, interactivity, causality, and privacy awareness [367], as well as prevent faulty behavior, weak points, unwanted biases, unfairness, and discrimination [23, 72]. Explanations are also used to provide better cognitive support and to improve the collaboration between humans and models [218].

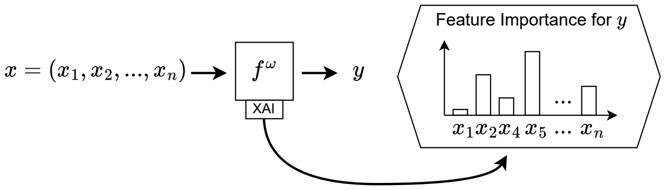


Figure 4.3: Conceptual diagram of an attribution technique that extracts feature importance from an input.

Within XAI, *interpretability* and *explainability* are two different concepts. According to Arrieta et al. [23], interpretability "*refers to the level of understanding of the models' internal decision process for a human observer*," while explainability is "*associated with the notion of explanation of the action or procedure taken by the model with the intent of clarifying its internal decision process*". Interpretability methods are mainly used when aiming to explain individual predictions of black-box ML models. In text classification, so-called *local explanation techniques* are widely adapted, which seek to attribute the importance or relevance of an input feature, such as a word, regarding a class c given the classifier f. Most commonly, local explanation techniques provide a scalar relevance value for each input feature, indicating its relevance concerning the predicted label y. Most commonly, each term of an input text is assigned a value indicating its positive or negative contribution to a particular class outcome. In contrast, a global explanation technique would try to understand the model as a whole. Figure 4.3 shows a conceptual diagram of an attribution method that explains a single prediction. Local explanations are typically modeled as an additional output to the final class prediction.

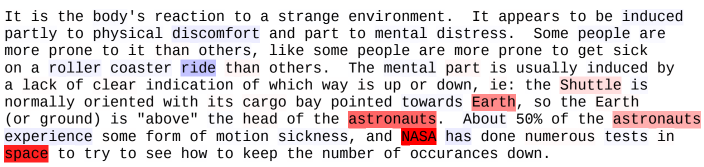

It is the body's reaction to a strange environment. It appears to be induced partly to physical discomfort and part to mental distress. Some people are more prone to it than others, like some people are more prone to get sick on a roller coaster ride than others. The mental part is usually induced by a lack of clear indication of which way is up or down, ie: the Shuttle is normally oriented with its cargo bay pointed towards Earth, so the Earth (or ground) is "above" the head of the astronauts. About 50% of the astronauts experience some form of motion sickness, and NASA has done numerous tests in space to try to see how to keep the number of occurances down.

Figure 4.4: A heat-map illustrates the explanation of a classifier regarding the label "space" [22]. Positive relevance is highlighted in red, and negative relevance is highlighted in blue.

Local explanations from text classifiers are most commonly visualized using heat-maps. They are employed to facilitate the analysis and understanding of explanations. A heat map is placed directly over the input text and highlights individual words according to a color gradient scheme, indicating the degree of relevance. Figure 4.4 illustrates an example of a local explanation. The relevance of each term is highlighted by a color gradient from blue to red. Highly relevant terms that contribute to the class label “space”, such as “NASA” or “astronauts”, are highlighted in red, while words that have a negative impact on this class label, such as “ride,” are highlighted in blue. Commonly used local explanation techniques include *Layer-wise Relevance Propagation* [27] and *Local Interpretable Model-agnostic Explanations* (LIME) [309].

4.4.3 Uncertainty Estimations

Applying a text classifier to a target dataset involves reducible and irreducible uncertainties [87]. Unfortunately, ML-based classifiers cannot adequately convey their struggles when processing an input instance [113]. Classification algorithms work like an input-output machine. A compatible input is always answered with a result. It does not check whether the input allows a plausible decision at all. In the worst case, a human will believe that a prediction is correct even though it is not. Therefore, it is essential to explicitly convey uncertainties to users, as otherwise uncertain results may be misinterpreted, leading to erroneous conclusions [278]. Within HiL applications, Dudley and Kristensson [98] outline that making users aware of classification uncertainties is a core solution principle for human-machine coordination since uncertainty information supports users to manage their expectations about a model's performance. Sacha et al. [319] contend that uncertainty awareness is critical in HiL approaches, as there seems to be a strong connection between user acceptance of model mistakes and the degree of uncertainty clarity.

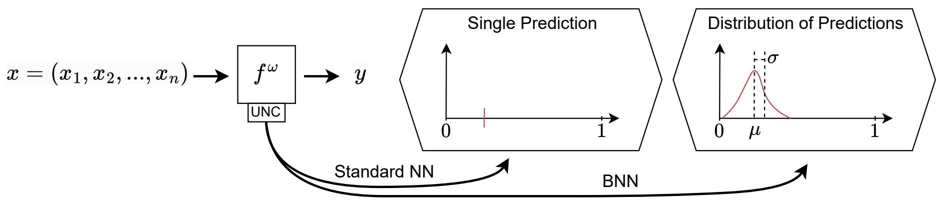


Figure 4.5: Conceptual diagram of an NN providing a class probability based on a single prediction or a distribution of predictions. These probabilities form the basis for uncertainty estimation.

The field of *uncertainty estimation* (Section 2.2.1) provides a practical solution for making the uncertainty inherent in classification outcomes explicit. As depicted in Figure 4.5, NNs provide uncertainty estimates as an additional output. Estimating the uncertainties of text classifiers has several advantages from a HiL point of view:

- Uncertainty estimates enable more trustworthy predictions by providing deeper insight into the reliability of model outputs. Knowing the uncertainty of a provided class label helps to make more informed decisions and to pay special attention to unreliable classification outcomes. Uncertainty measures can reduce the risk of relying on incorrect model behavior [148, 372].
- Uncertainties provide an additional measure of the classification performance. They allow for more informed decisions in model selection [31], as an overall low uncertainty results in more accurate predictions. Additionally, high uncertainties indicate that a model has not acquired enough information to make accurate predictions. In this case, more training data may be necessary [221].
- Uncertainty facilitates protection against inputs that are likely to be misclassified [148, 162]. Common examples are out-of-distribution instances, to which classifiers generally do not generalize well [148].

4.5 Design Patterns for HiL Systems

While the concept behind HiL is promising, its research space is rather convoluted, and its operationalizability and applicability in software development practice are unexplored. Moreover, the design space requires a careful, systematic consideration and guidance. First, human assistance in creating and using ML models is usually costly and should be carefully designed and minimized. Second, re-training ML models can be very expensive in certain cases, with a significant carbon footprint. Third, both models and humans can make (different types of) mistakes, which requires a careful trade-off discussion and decision that fits the use-cases and domains at hand.

This thesis proposes a catalog of 12 patterns for designing ML-based systems with HiL to guide developers in choosing and adapting HiL solutions that fit their needs and raise awareness about HiL in general. While HiL has been around for a while as a paradigm for designing and evaluating ML models, we focus on the software engineering perspective, providing recurring, reusable HiL solutions for designing ML-based systems given certain contexts. We define a “design pattern for ML with HiL” as any pattern that directly or indirectly addresses the learning behavior or outcome of an ML-based system through human feedback. Like other design patterns, our patterns are general solutions and best practices to a common and recurring problem faced by many practitioners. Our pattern catalog provides a common vocabulary to better communicate the

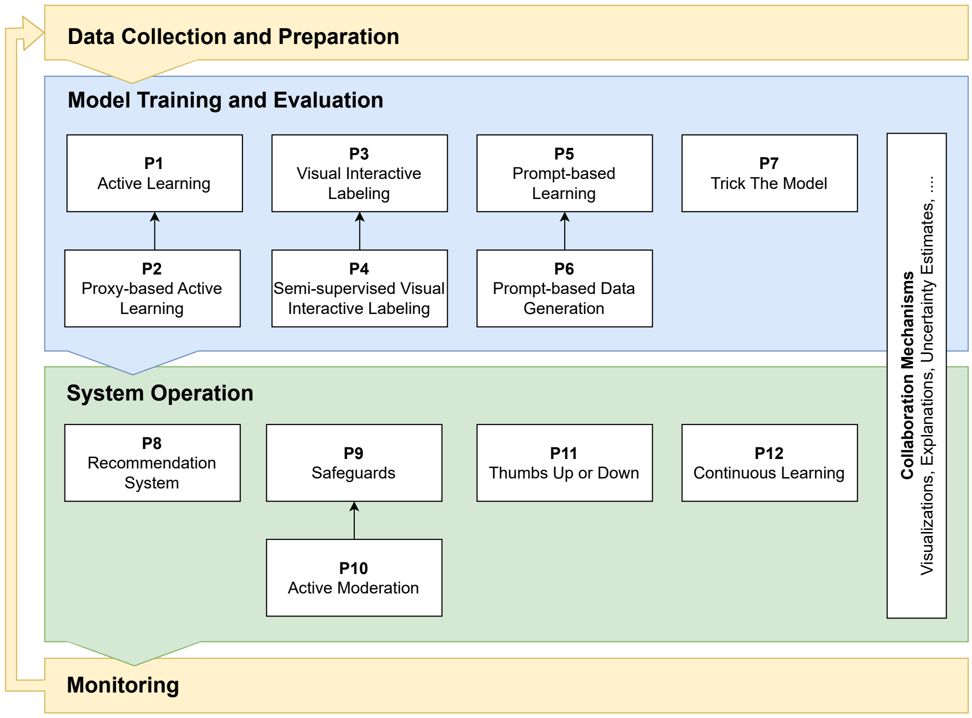


Figure 4.6: Our proposed HiL pattern catalog includes seven training and five operation patterns. Arrows between the patterns indicate dependencies. Collaboration mechanisms are orthogonal to the patterns and enhance human-model interaction.

proven solutions for designing a HiL system. While software engineering patterns have been proposed for designing ML systems in general [209, 236, 384], there is a lack of design knowledge on how to effectively design and deploy HiL solutions.

This thesis compiles a catalog of design patterns (as shown in Figure 4.6) to guide developers in selecting and implementing suitable HiL solutions. Our catalog considers critical requirements such as the cost of human involvement and model re-training. The catalog is not complete and can be expanded. We have focused on the most popular patterns with the greatest practical utility and clarity. The identified patterns are divided into two groups according to their purpose: training patterns, which are used to make the initial training much more effective, and operation patterns, which are applied during deployment to further improve the applicability of the model. Our pattern catalog includes seven training and five operation patterns.

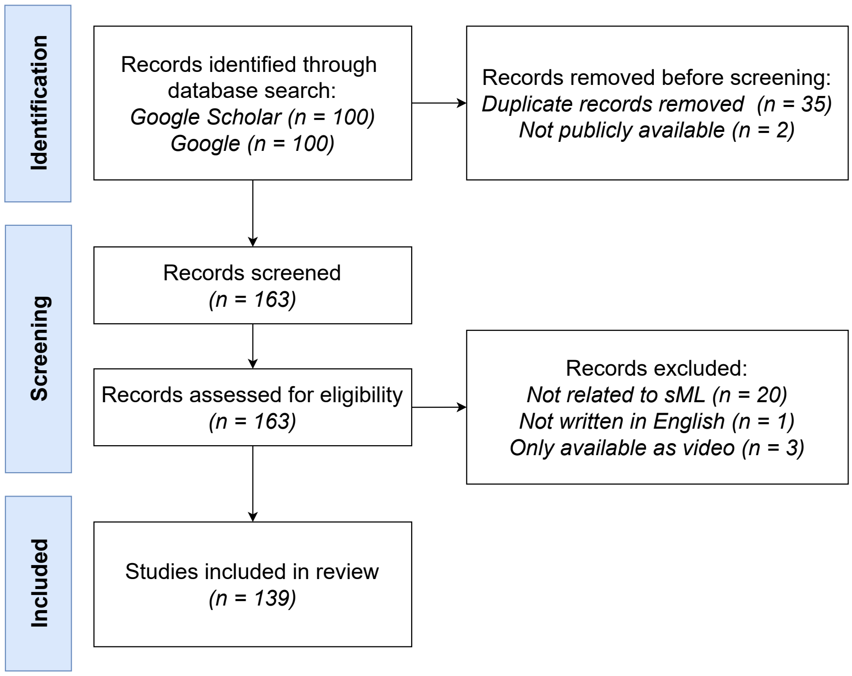


Figure 4.7: A flow diagram illustrating the record selection process according to the PRISMA 2020 statement.

4.5.1 Research Methodology

We conduct a literature review to investigate and identify best practices for the design and deployment of HiL systems. The selection and reporting of our review are inspired by the guidelines outlined in the PRISMA (*Preferred Reporting Items for Systematic Reviews and Meta-Analyses*) statement [281]. PRISMA provides a structured approach to reviewing and synthesizing existing literature and ensures a comprehensive and transparent methodology.

The objectives of the literature review include synthesizing existing knowledge and providing engineering insights for practitioners and researchers. The guiding research question of our study is: "*What are the prevailing best practices for employing or implementing the HiL approach in ML systems?*" To ensure the relevance and rigor of the review, we define specific inclusion and exclusion criteria. Included studies or applications must concentrate on HiL solutions or approaches in the context of supervised ML. We have also included articles that discuss conceptual HiL approaches without providing a concrete implementation. Additionally, studies had to be written in English and had to be open-access. The literature review includes peer-reviewed journals, conference papers, and gray literature, such as non-peer-reviewed research papers, blog posts, and technical reports.

Figure 4.7 depicts the selection process of the included records. A comprehensive search strategy was developed to identify relevant literature. We queried Google[1] and Google Scholar[2] for papers and applications of HiL in the context of supervised ML, using the query string: *'human-in-the-loop AND ("machine learning" OR classification OR regression)'*. For both search engines, the first 100 results were considered. A two-stage screening process followed the study identification process. Initially, for research papers and articles, we screened the titles, abstracts, and introductions based on the predefined inclusion and exclusion criteria. For blog posts we, screened the entire text. Subsequently, the full-text is reviewed to assess their eligibility for inclusion. We included a total of 139 records. Data synthesis involved summarizing and organizing the findings from the selected records. Common themes, patterns, and applications related to best practices for HiL in ML systems were identified and grouped. The grouping did not use predefined classes. We distinguished between how humans are integrated into the ML process, the goal of human involvement, and where an interaction occurs in the ML pipeline. The synthesis process aimed to provide a first overview of the literature and offers insights into the current practices of HiL in ML systems.

Furthermore, we intend to combine the results of the literature review with our own experience in the research and development of HiL systems [11, 12, 14–17]. Due to the limited scope of our literature review, it is likely to miss approaches that are considerable established HiL patterns. We intend to incorporate our own experience into the catalog to ensure a more comprehensive and useful coverage of the key areas of the ML process.

4.5.2 Study Findings

Our findings reveal that the research area of HiL is rather convoluted, and there is a lack of common knowledge on how to design HiL systems effectively. While there are several literature reviews on HiL, the included studies focus on single aspects of HiL, mainly to collect data for learning. Typically, the HiL approach is vaguely defined and nebulous in its characterization. Generally, HiL refers to some rapid, focused, and incremental cycles of interaction between users and learning systems [8], leading to ambiguity and lack of clarity in the design of HiL systems. While many HiL solutions have been developed and proposed in the literature, there is a lack of structured approaches and guidelines on how to systematically address common problems of ML systems via HiL.

[1] `https://www.google.com/`
[2] `https://scholar.google.de/`

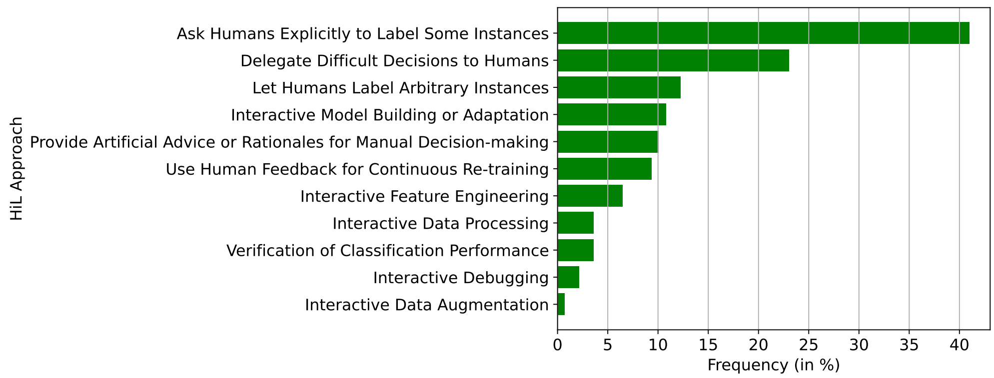


Figure 4.8: Frequency of the HiL approaches extracted from the literature.

Figure 4.8 displays an overview of the identified HiL approaches sorted by frequency. In the literature we reviewed, the majority of HiL approaches focus on the task of data labeling. The most frequently described HiL approach is the effective collection of labeled data for the initial training of ML models (41%). Here, human labelers act as oracles, responsible for labeling a sequence of artificially selected data instances during the initial training. When done effectively, the time and effort needed to train ML models is significantly reduced. Another common HiL approach is to rely on humans to oversee artificial predictions in case model predictions are too unreliable for automated deployment (23%). Here, we see the human as an oracle, which is involved in the deployment phase of an ML model. The goal is to reduce the number of incorrect model outcomes when the model is used in deployment. The dominance of these two approaches, which account for more than 64% of the analyzed samples, indicates a lack of diversity in the current literature on HiL. The next most common approach is to have humans label arbitrarily identified instances selected by themselves (12%), rather than directly asking humans for instances to be labeled. Using human feedback from labeling or human interventions to re-train the model from time to time was described or outlined in 9% of the reviewed work.

Another significant portion of the studies use the HiL approach to integrate humans more tightly and quickly into individual steps of traditional ML pipelines. These include iterative data processing (4%), feature engineering (6%), model building and adaptation (11%), as well as ML debugging (2%), and data augmentation (1%). For example, humans may be asked to modify the logical rules on which the classifier is built, suggest features, or manually remove stop words. Such approaches are usually highly dependent on the actual task and model being used, and are rarely solvable by domain experts. Furthermore, NLP models are typically based on neural models or learned feature representations that are difficult to adapt interactively, even for ML experts. Notably, more recent approaches, such as LLMs, are not mentioned in the identified records.

In the next step, we identified three main groups of HiL approaches that are worth distinguishing. The first group includes HiL approaches that rely on technical knowledge or ML expertise to provide valuable feedback. These approaches include activities such as interactive model building, feature engineering, and data processing. However, as previously outlined, HiL solutions primarily target domain experts rather than ML experts and engineers. Domain experts provide valuable knowledge that machines cannot provide. This is particularly important since humans are intended to support tasks that ML models struggle to perform. Therefore, we did not include them in our catalog. The second and third groups cover HiL approaches according to the step where human interactions take place in the ML pipeline. We distinguish between training and operation patterns. This structure is also reflected in the final pattern catalog. Based on these findings, we have extracted concrete patterns to facilitate the development and design of HiL systems. All patterns are described in a consistent scheme. Since the reviewed records did not cover some recent HiL approaches, in particular those using decoder-based LLMs to mimic supervised ML, we included some patterns based on our own experiments. In total, 12 concrete HiL patterns for the design and development of ML systems were extracted. We outline variations for certain patterns to suggest alternative implementations and explicitly focus on domain experts as humans and the task of supervised ML.

4.5.3 Training Patterns

Modern supervised ML models require a large amount of labeled data to be properly trained. However, labeled data for specific real-world applications (beyond common tasks and scientific benchmarks) is typically sparse and very expensive to create. The following section outlines general patterns for the effective HiL training of ML models.

P1: Active Learning [332]

Goal: Minimize the labeling cost (number of labels assigned) by carefully selecting which instances are worth labeling manually. The objective is to create a minimal training dataset capable of training a highly accurate ML model.
Human Role: Label a sequence of automatically selected data instances.
Problem: ML models require numerous high-quality training examples to re-

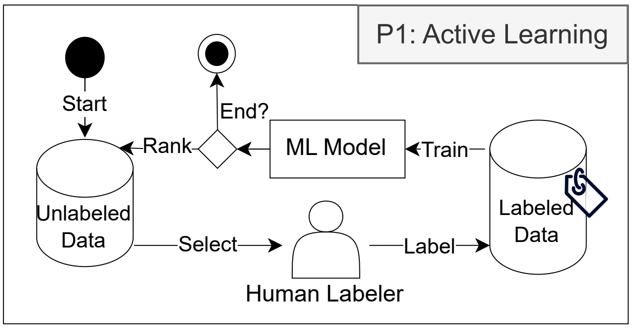


Figure 4.9: The Active Learning pipeline. During training, unlabeled data instances are to be labeled by a human oracle. The labeled instances are then added to the training dataset. At the end of each training iteration, the model is re-trained and possibly improved.

liably recognize patterns in unseen data. Since training examples are typically sparse or unavailable, unlabeled instances must be manually labeled. However, labeling is a labor- and cost-intensive task that is impractical on a large scale. It is desirable to use human effort most effectively during the labeling process of the initial training data.

Structure: In Active Learning [221, 332], a model actively selects its own training data from a typically large corpus of unlabeled data instances. It has been shown that a careful selection of highly informative and expressive instances can drastically reduce the amount of training data needed to achieve a given level of performance [221]. Thus, Active Learning can heavily reduce the cost of the labeling process. The process of Active Learning consists of the following steps (Figure 4.9): First, potential unlabeled training instances are ranked according to their expected contribution to a model's learning behavior. The k-highest ranked instances are then labeled by a human oracle and added to the training data. The model is then re-trained until a stopping criterion is met, i.e., the labeling budget is exhausted or the model's classification performance is good enough.

Advantage:

- Saves human resources and time during development and (pre-) deployment.
- An easy-to-implement and straightforward process.

Challenges:

- Frequent re-training of ML models causes high computational cost.
- High degree of dependency between the training data and the trained model.
- Requires low model latency to maintain user experience.

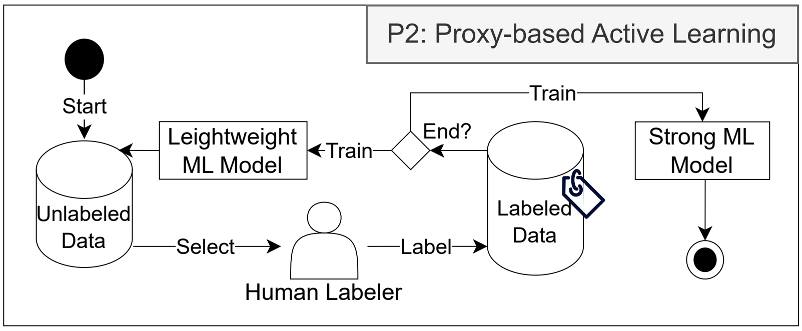


Figure 4.10: The Proxy-based Active Learning pipeline uses a lightweight model, typically with low latency, for Active Learning. The gathered training data is then used to train a much more powerful model.

- Labeling a model-selected sequence of instances tends to be a boring and tedious task at scale.

When to apply: There is a lack of labeled examples, but a large amount of unlabeled data is available. An extensive user interface is not required.
Example Implementation: ActiveAnno [388] is a versatile text annotation tool that implements Active Learning. Its user interface is simplistic, displaying only the instances requiring manual annotation, while offering the functionality to attach annotations. Paladin [272] also allows labelers to configure the Active Learning process by choosing the data selection strategy or batch size. The tool can also select which of several potential labelers to query based on their estimated confidence level.

P2: Proxy-based Active Learning [71]

Dependency: Variation of the Active Learning pattern (P1).
Goal: Maintain an exceptional user experience with respect to model latency throughout the Active Learning process, while deploying a powerful model that maximizes performance.
Human Role: Label a sequence of automatically selected data instances.
Problem: When high classification performance is required, complex models are often preferred due to their ability to yield significantly higher classification performance than simpler models. However, these highly complex models often result in significant latency, particularly during the training phase. Active Learning requires a re-training step after each iteration, which results in significant waiting time for human labelers to provide feedback. A high latency adversely impacts the user experience [93, 361]. Keeping the latency of the ML model low is critical to preserve the user experience of Active Learning.

Structure: Proxy-based Active Learning uses two models to maintain a high user experience during labeling the initial training data, while maximizing the classification performance during deployment (Figure 4.10). First, a lightweight model with very low latency is used to select training data according to the Active Learning pattern. The low computational latency ensures a high user experience. Then, the collected training data is reused to train a more complex and, ideally, more powerful model. The latter model is then deployed to maximize operational efficiency.

Advantage:

- Maintains a high level of user experience by facilitating rapid interaction cycles during labeling.
- Saves computational resources by performing frequent re-training with a lightweight model. The complex model is only trained once.

Challenges:

- Training a model on samples not chosen by the model itself is likely to lead to a loss of classification performance compared to employing the same model for both data selection and training [362].
- Increased effort to develop and maintain two ML models.

When to apply: Maintain user experience during the labeling process when the target model has too much latency, making conventional Active Learning impractical.

Example Implementation: Proxy-based Active Learning is a technical variation of the Active Learning pattern first discussed by Tomanek and Morik [362]. Chapter 7 presents a further investigation into Proxy-based Active Learning for text classification.

P3: Visual Interactive Labeling [328]

Goal: Allow human labelers to visualize and explore an unlabeled dataset, enabling the human-centric selection of appropriate training data.

Human Role: Label self-selected data instances.

Motivation: When an ML model selects its own training data (as in Active Learning), there is a risk of selecting redundant information or outliers, which reduces the effectiveness of the labeling process [154]. In addition, a machine-centric selection does not incorporate human skills and expertise into the data selection process. As a result, much potential remains untapped when experts of the problem-domain are engaged as labelers.

Structure: Enabling a human-centered selection of training instances requires some form of visual environment that facilitates analytical reasoning, i.e., human

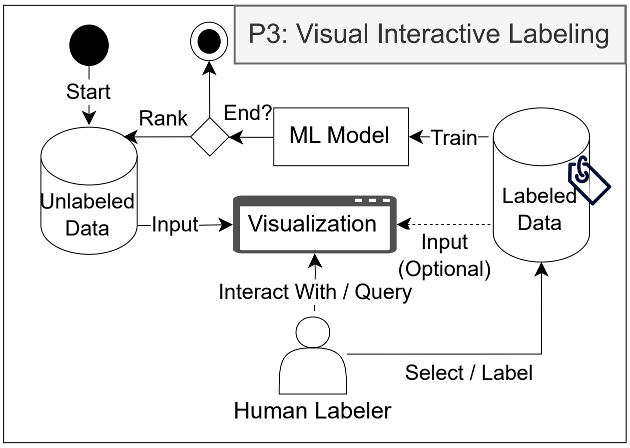


Figure 4.11: In Visual Interactive Labeling, humans are responsible for selecting instances worth labeling for re-training. Labelers are provided with some data visualization to explore and label instances to facilitate human-centric data selection strategies.

perception of patterns and structures in the data. Visual Interactive Labeling adapts a visual environment to enable a human-centric annotation process (Figure 4.11). Training data is selected and annotated directly by humans. As with Active Learning, the model is periodically re-trained, and the visualization is updated.

Advantage:

- A more active human role could increase labeling efficiency compared to Active Learning.
- Interactive visualization and exploration empower labelers to gain insights and knowledge about their data.
- Enables a continuous flow of user interactions without waiting times (latency).

Challenges:

- Requires a visual interface to facilitate the labeling process.
- May require trained humans to make sense of the visualization.
- Visualizations become cluttered at scale.

When to apply: More flexibility during the data selection is needed compared to Active Learning. Requires a visual labeling tool and an expert who can perform appropriate user-centered data selection.

Example Implementation:

Seifert and Granitzer [328] propose one of the first Visual Interactive Labeling tools. They use a radar visualization where all unlabeled instances are displayed based on the confidence of the model. Alto [291] is another Visual Interactive

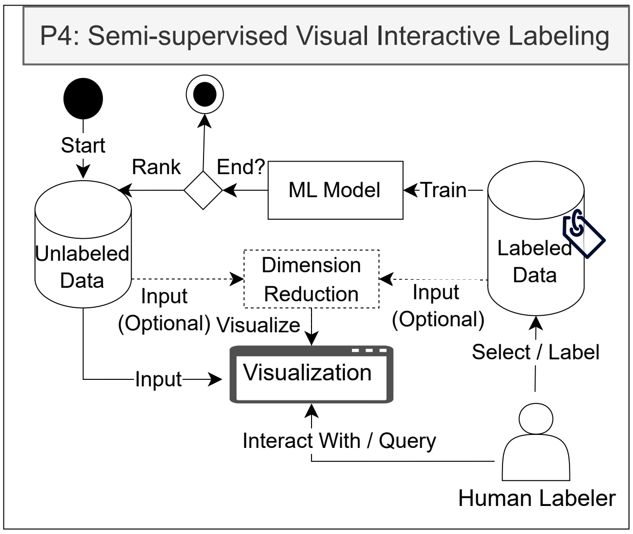


Figure 4.12: Semi-supervised Visual Interactive Labeling involves using a dimension reduction model to generate a latent space visualization that facilitates the labeling process.

Labeling tool that aims to provide labelers with global knowledge about the dataset to be labeled. A topic model is used to provide an overview of the labels to be created. Furthermore, the tool suggests a list of examples to the user. Users can then select instances to label based on their preferences.

P4: Semi-supervised Visual Interactive Labeling [34]

Dependency: Variation of the Visual Interactive Labeling pattern (P3).
Goal: Semantic embedding spaces provide valuable insights into the data, enabling an effective and user-centric selection of training instances.
Human Role: Label self-selected data instances within a semantic space.
Motivation: Human-centric labeling approaches rely heavily on meaning and insightful representations to best use human capabilities. However, manually extracting and finding associations and structures for labeling large datasets is challenging and needs to be supported. Dimension reduction [102] facilitates labeling by making hidden semantics between data instances explicit. Dimension reduction is a tool for extracting low-dimensional representations of high-dimensional data while preserving the semantic relationship between all data instances. It can make the data instances much easier to visualize while preserving their semantic relationships.
Structure: Semi-supervised Visual Interactive Labeling uses a dimension reduction model to compute a low-dimensional and easily visualized representation of typically high-dimensional data (Figure 4.12). The low-dimensional data is

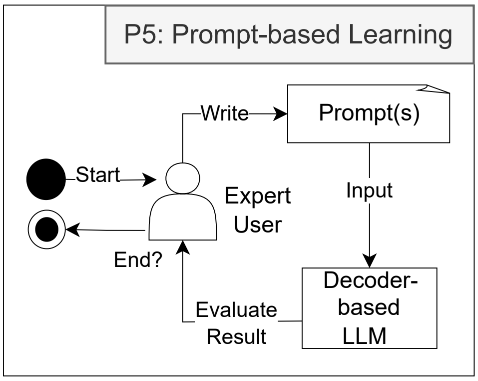


Figure 4.13: The Prompt-based Learning pipeline. Humans translate the task to be solved into prompts, which are then answered by a decoder-based LLM. The process may involve multiple iterations of prompt engineering to appropriately address and solve the task.

then visualized using a scatter plot. The closer two data points are, the more semantically similar they are, and vice versa. Labeling is supported because neighboring instances are likely to have the same label. In addition, exploring and understanding the plotted data facilitates the generation of knowledge about the data, further improving the labeling process.

Advantage:

- A more active role of humans could increase labeling efficiency compared to Active Learning.
- Emphasis on human insight and knowledge generation from data.
- Semantic similarities reveal patterns and structures in the data.

Challenges:

- Requires an expert in the field to interpret the scatter plot and apply their knowledge to effectively accomplish the labeling process.
- Additional effort of training a dimension reduction model.
- Requires a visual interface to facilitate the labeling process.
- Scatter plots become visually cluttered at scale.

When to apply: There are semantically meaningful topics hidden in the data and experienced labelers available to support human-centric labeling.

Example Implementation: A typical implementation of Semi-supervised Visual Interactive Labeling is AILA [69], which stands for *Attentive Interactive Labeling Assistant*. The system consists of an embedding view that lays out documents using *t-distributed Stochastic Neighbor Embedding* (t-SNE) [244]. Users can then use the embedding view to discover and select instances for labeling.

P5: Prompt-based Learning [161]

Goal: Translate the text classification task into a masked language modeling problem that requires no training data.
Human Role: Prompting a decoder-based LLM to solve a specific task directly.
Motivation: Supervised text classifiers require labeled training data in order to learn patterns and structures in new data. While labeled data is typically the bottleneck during training, unlabeled data may also be unavailable and difficult to obtain. In this case, there is no data to label or learn from.
Structure: Prompt-based learning for text classification relies on decoder-based LLMs that model the probability of a text $p(x)$ rather than the probability of a label assignment $p(y = c|x)$ (Figure 4.13). The text classification problem is transformed into a masked language modeling problem. Prompt-based learning uses the general knowledge acquired by an LLM to solve a specific task without additional training or parameter tuning. No supervised learning is needed. The model is tuned during inference with a task-specific prompt. A prompt is a piece of text containing a semantic description of the classification task designed by a human. An input text x to be classified is wrapped into a semantic description of the classification task and passed to the model. For instance, the input x is wrapped into a template such as "[x] It was a [MASK] movie!". The prediction is made based on the probability that a class-related word, such as "great" for "sentiment positive" or "bad" for "sentiment negative" will be filled in the [MASK].
Advantages:

- Does not require manual data labeling.

Challenges:

- Requires the use of an decoder-based LLM that consumes significant computational resources or is accessible only through paid black-box APIs.
- May require multiple iterations to create, refine, and analyze prompts, requiring additional knowledge.
- Results could be misleading (hallucination [170]).

When to Apply: There is a lack of unlabeled data to perform any form of supervised ML.
Example Implementation: Arco et al. [19] propose a prompt-based learning approach to hate speech detection. They utilize the template: *"<text> This text is <verbalizers>"* to prompt a decoder-based LLM given some text. After prompting the LLM, they assess the likelihood of hateful and non-hateful verbalizers and select the most probable completion as the final prediction.

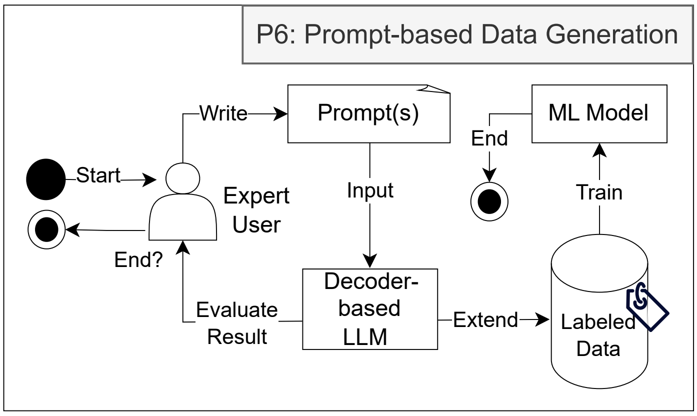


Figure 4.14: The Prompt-based Data Generation pipeline. A decoder-based LLM is prompted to generate artificial training examples. These are then used to train a supervised ML model.

P6: Prompt-based Data Generation [398]

Dependency: Variation of the Prompt-based Learning pattern (P3).
Goal: Leveraging an LLM to generate data for training a distinct ML model.
Human Role: Prompting an LLM to generate high-quality training data.
Motivation: While labeled data is typically the bottleneck during training, unlabeled data may also be unavailable and difficult to obtain. In this case, there is no or not enough data to learn from, so a supervised text classifier cannot be developed.
Structure: An LLM is prompted to generate a set of labeled text instances given some context (Figure 4.14). The generated data is then used to train a supervised ML model. The LLM is tasked with probabilistically filling in gaps or continuing a user-provided starting prompt according to a desired outcome. For example, an LLM might be prompted to fill the gap in the sentence "Overall, it was a [_] film." to satisfy a variety of sentiment levels. A secondary task-specific supervised learning model is then trained on the generated data, extending the training data.
Advantages:

- Does not require manual data labeling.

Challenges:

- LLMs require significant computational resources or are only accessible through paid APIs.
- May require multiple iterations to create, refine, and analyze prompts, requiring additional knowledge.
- It is challenging to align LLMs with the actual target domain.

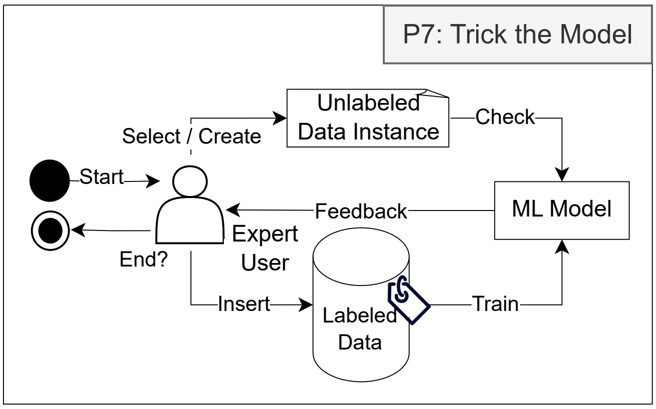


Figure 4.15: In the Trick the Model pattern, humans iteratively select or modify data instances against which the behavior of the model is checked. If incorrect behavior is detected, the gap in the model must be closed by providing additional learning examples.

- Synthetic data tends to be less informative and redundant.
- Requires an LLM and a supervised ML model.

When to Apply: Unlabeled data is not available to the extent needed to train an appropriate ML model.

Example Implementation: ZeroGen [398] is a simplistic method for generating a labeled dataset from scratch. First, a decoder-based LLM is prompted to generate labeled text instances in an unsupervised manner. The generated dataset is then used to train and deploy a lightweight target model.

P7: Trick the Model [26]

Goal: Expose model errors to make the model more reliable against a variety of inputs.

Human Role: Test the model (and trick it) with inputs that cause incorrect behavior.

Motivation: Models sometimes poorly transfer learned patterns into real-world data. Implementing a model without runtime errors does not mean that it will always work as intended. Even after acquiring a reasonable number of training examples, cases may occur where a model provides unsatisfactory outcomes. Errors may not be in the software but hidden in the learned parametrization caused by poor data quality. Insufficient training data leads to blind spots, biases, or vulnerability to adversarial attacks. Additional data is needed to fill these gaps and make the use of the model much more reliable and secure.

Structure: Trick the Model involves providing additional training instances to make the model much more reliable and efficient (Figure 4.15). First, humans interactively evaluate how models behave on manually constructed or slightly modified inputs (e.g. cropped images or edited text) to detect unintended behavior, such as misclassifications. The focus is especially on inputs, where the model is highly confident in its prediction, but is actually wrong. When unintended behavior is found, the goal is to close the gap by collecting or creating additional training examples that address the misconception. Tricking a model requires humans to understand what a model has learned and when it is likely to fail. Since it is difficult to impossible for a human to understand how a model makes decisions, an explanation mechanism that provides human-understandable insights into the behavior of the model can help, e.g. showing which features of the input contributed most to a class-specific decision.

Advantage:

- Identifies and reduces model misbehavior prior to deployment.
- Demonstrates the robustness of an ML model.
- Informs users about the adversarial training process and its purpose, which can increase transparency and foster trust.

Challenges:

- Requires additional computational resources and time for training and testing.
- Nondeterminism in model optimization can lead to unintended side effects.
- Compelling adversarial examples require a deep understanding of the weaknesses and decision boundaries of an ML model, which can be complex and difficult to characterize.
- While evidence of misconduct can be provided, conclusive evidence of its absence cannot.
- Requires a certain degree of transparency to comprehend model outcomes.

When to apply: The model performs well overall, but the data tends to be unrepresentative. There is a high risk of misbehavior, and the model is too uncertain on some parts of the data to be used operationally.

Example Implementation: TextAttack [261] is a tool for securing and testing models against adversarial examples. A so-called "attack" consists of four main components: a goal function, a set of constraints, a transformation, and a search method. The objective of the attack is to modify an input text to satisfy the goal function (determining the success of the attack) while ensuring that the perturbation adheres to the specified constraints (such as grammar or semantic similarity). A search method is utilized to discover a sequence of transformations leading to a successful adversarial example. Attacks are also employed for data augmentation and adversarial training.

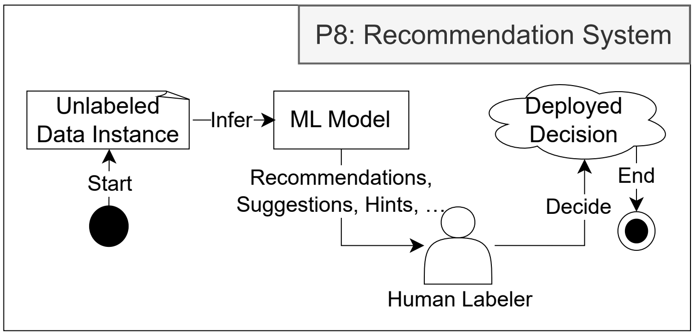


Figure 4.16: The process of Recommendation System. Each instance is initially passed to the model. The model predicts the instances and provides some kind of recommendation to facilitate human decision-making. The human makes the final decision.

4.5.4 Operational Patterns

HiL processing is not limited to the training phase but can also be effectively applied throughout the entire life cycle of a model. The following section focuses on HiL operational patterns for ML models.

P8: Recommendation System

Goal: Enhance the quality and efficiency of decisions.
Motivation: Many real-world application domains are high-stakes tasks that involve high-impact but difficult decisions, such as medical diagnosis [193]. When the consequences of errors are significant, a purely ML-based approach usually reaches its limits due to a general lack of user trust, correctness, and legal responsibility. In this scenario, a domain expert (e.g., a doctor) is responsible for labeling some data (e.g., disease detection and classification). However, manual decision-making can be corrupted by human biases, unintentional errors, natural fuzziness and ambiguity, or take much time.
Structure: To actively support a domain expert in making critical decisions, ML models can provide an initial suggestion or recommendation for decision-making (Figure 4.16). In addition to a simple label, a suggestion can include comprehensible evidence as to why an ML-based approach would select a specific outcome or how certain the prediction is. The support of the ML model provides a second opinion and reduces the risk of humans overlooking essential things.
Advantage:

- Improves the accuracy of decision-making by providing artificial evidence.
- Improves the efficiency by quickly receiving recommendations and hints.

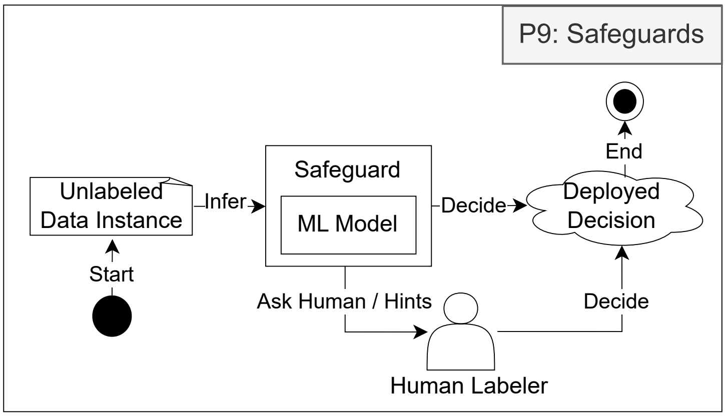


Figure 4.17: In the Safeguards pattern, an ML model is secured by a set of rules that prevent it from making predictions that are known to be very difficult and likely to be wrong. These cases are delegated to a human labeler to provide more reliable labels.

Challenges:

- Requires the model to provide evidence as to why a particular recommendation might be favorable.
- Risk of information (recommendation) overload or blindly trusting false evidence.
- Does not scale well to large workloads since a human expert still makes every decision.

When to apply: Tasks with a high level of difficulty and significant consequences of failure that typically must be accomplished by humans.

Example Implementation: The Recommendation System pattern is most frequently utilized within high-stakes tasks such as medical healthcare. A typical use-case is to assist healthcare professionals in making critical diagnoses or treatment decisions. Another common use-case is disaster management. For example, the RADAR system proposed by Sengupta et al. [331] aims to support complex decision-making in the domain of firefighting. Human commanders are supported by adaptive plans to respond to a fire. Decision Support systems can be easily adapted to text classification tasks, such as fraud detection.

P9: Safeguards [197]

Goal: Ensure highly reliable decisions by using a safety wrapper, i.e., deterministic rules, to deal with the inherent uncertainty of model results.

Motivation: Due to the complexity and statistical nature of ML models, there can be no 100% guarantee of their correctness [356]. Even a trained and fine-tuned ML model may still not be accurate enough to reach the level of clas-

sification performance required to use the model productively. Adding more training examples does not always lead to top classification performance, as the achievable performance converges to a maximum as the size of the training data increases [114]. Model predictions are susceptible to uncertainties arising from various sources, including noisy data or the inherent complexity of the problem [87]. To ensure reliable deployment, it is crucial to address these uncertainties, which can introduce variability and affect prediction performance.
Structure: A valuable step in achieving more reliable predictions is to encapsulate the model's functionality with a safety layer, such as verifiable rules (Figure 4.17). Inputs that are unlikely to be handled correctly by the ML model are filtered out. Safeguards aim to mitigate the consequences of each decision, i.e. they refuse to decide on ambiguous situations. Rejected instances are then assigned to human labelers to provide a more reliable decision. A safeguard can be a deterministic, hand-crafted safety rule or a rejection model.
Advantages:

- Gracefully handles failure cases.
- Provides a higher level of reliability than a purely automated ML model.

Challenges:

- Safeguards themselves poses uncertainty and vagueness.
- Safety rules are difficult to define.
- A static mechanism does not adapt to changing data distributions and new concepts.

Example Implementation: Link et al. [229] propose a HiL moderation approach for social media content, including safeguards. The tool supports human moderators in correcting artificial predictions by assigning instances to human labelers when they do not contain enough information for automated classification. Various filtering rules are used to minimize the human resources required for manual moderation.

P10: Active Moderation [11]

Dependency: Variation of the Safeguards pattern (P9).
Goal: Improve or maintain a required level of classification performance that a model alone cannot deliver during deployment.
Motivation: Defining safeguards for complex feature spaces is challenging. Additionally, the safeguards themselves can introduce additional uncertainty.

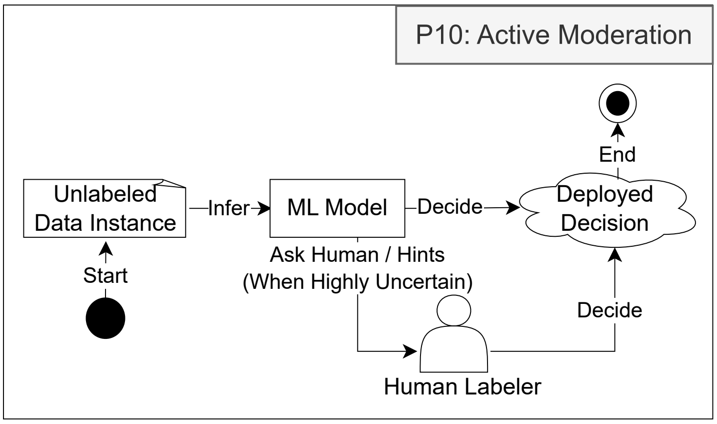


Figure 4.18: The Active Moderation pattern uses the prediction uncertainty of the actual models in operation to decide whether a human is needed to provide more reliable decisions.

They do not account for the statistical vagueness of classification models, nor do they easily adapt to new and changing data. Furthermore, abstaining from unreliable predictions is a binary decision, making it difficult to define abstention thresholds. There is a knowledge gap between engineering external safety layers and capturing the true difficulties and uncertainties of models.

Structure: Active Moderation aims to improve the classification performance by delegating likely to be wrong model outcomes to a human oracle, while limiting human labor (Figure 4.18). To prevent unreliable predictions, Active Moderation relies on the prediction uncertainty of the model used. The idea is to manually decide only those cases for which a model cannot provide reliable suggestions due to its high uncertainty. Simple predictions, on the other hand, do not require manual verification because they are close to being correct. For each prediction, it is determined whether the prediction is likely to be correct (confident) or not (uncertain). If a prediction is uncertain, a human needs to give a reliable judgment instead. Since ML models generally do not make statements about their own correctness, special mechanisms (uncertainty estimation and quantification) are needed to quantify the reliability of the prediction.

Advantage:

- Gracefully handles failure cases.
- Provides a higher level of reliability than a pure ML-based model.
- Relies on the statistical uncertainty of the actual ML model itself. No other models or external mechanisms are required.
- Provides a dynamic and data-driven approach that adapts to changing data.

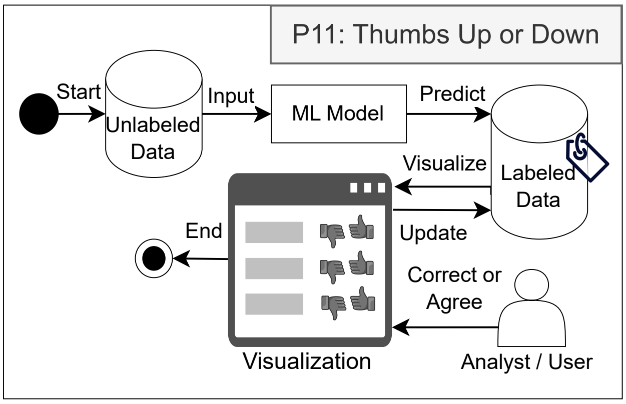


Figure 4.19: During the Thumbs Up or Down process, humans are empowered to correct or approve artificial decisions to correct potential mislabeling in the data.

Challenges:

- Uncertainty estimates cannot detect all model misbehavior [26]. Only those misclassifications that receive high uncertainty scores.
- Achieving near-perfect classification performance requires considerable manual effort.
- Requires high-quality uncertainty estimates.

When to apply: When already trained ML models do not provide the level of classification performance required to solve a task appropriately.

Example Implementation: Hendrycks and Gimpel [148] were among the first to demonstrate that prediction uncertainties are capable of detecting potential misclassifications. The Active Moderation approach, especially its cost-effectiveness, is studied in more detail in Chapters 5 and 6.

P11: Thumbs Up or Down [17]

Goal: Enable fast deployment by allowing humans to correct incorrect model outcomes on the fly when using ML systems.

Motivation: Deploying a highly accurate ML model is a lengthy process, as training data is sparse and can change over time. Cases where ML-based predictions are incorrect can be expected, especially in the early stages of deployment. Errors identified by a human during the deployment of an ML system should therefore be correctable.

Structure: ML-based systems should allow users to correct the outcome of the ML model in case they are incorrect (Figure 4.19). The interface has to provide a mechanism for correcting model outcomes. A common approach is to provide a button to overwrite, re-label, or correct all model predictions.

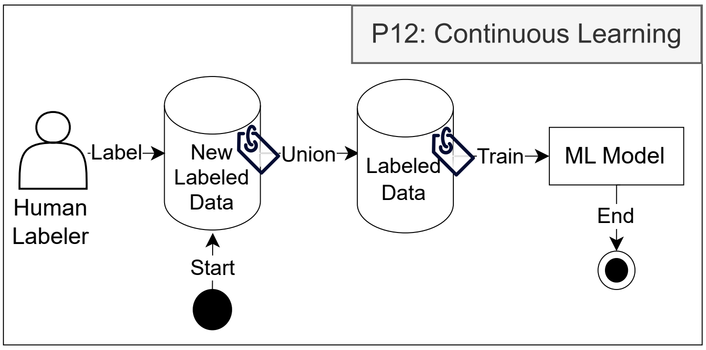


Figure 4.20: In the Continuous Learning pattern, an ML model is continuously retrained as additional labeled data becomes available.

Advantage:

- Erroneous model behavior can be corrected immediately, making the overall analysis more accurate.

Challenges:

- Analysts may be distracted from the actual analysis by an increased need for correction.
- Allowing users to provide feedback may reduce their trust in the system and their perception of the system's classification performance [156].

When to apply: ML systems for recommendations or sensemaking of individual predictions.

Example Implementation: OpenReq [351] is an analytics tool for collecting, processing and analyzing user feedback from online platforms such as marketplaces. It offers an enriched visual interface to facilitate an exploratory data analysis process. The tool allows the user to disagree with the classification results during the analysis, but only in cases where the model itself is uncertain.

P12: Continuous Learning [89]

Goal: Update the ML model on a regular basis to ensure that it continues to provide accurate and reliable predictions.

Motivation: Real-world data is highly dynamic and subject to change. For ML models, it is critical that the training data match the current state of the environment in which a system is deployed. If a model is applied to new data that is significantly different from the training data, the model will struggle to recognize patterns it has not seen before, leading to incorrect results.

Structure: Deployed ML models need to be updated with data that reflects their current environment to maintain their effectiveness. Continuous Learning aims to retrain a model on a regular basis when additional training data is available, e.g. daily or when a certain amount of additional labeled data is available (Figure 4.20). In this way, the model learns new patterns and trends and continuously adapts to changes in the environment, i.e. new terms, slang, image resolutions, laws, etc. The training data is extended with labeled examples that arise during the use of the systems.

Advantage:

- Keeps the model up-to-date and prevents it from decaying over time.

Challenges:

- Continuously re-training a model is computationally expensive.
- Requires ongoing monitoring and quality assessment.

Example Implementation: The Journalist-in-the-Loop system [182] enables journalists to provide annotated feedback to the system during their daily work. Its objective is to classify rumors while allowing journalists to improve predictions as they use them. The feedback is used to continuously re-train the model and keep it up to date.

4.6 Discussion

We emphasize the need to involve domain experts in the text classification process to increase or even enable the applicability of ML-based text classification. The HiL paradigm holds significant promise for enhancing not only the training but also the deployment of text classifiers. While HiL aims to overcome the significant applicability challenges of ML models, it introduces new challenges that require careful consideration. The high dependence on humans and the associated high costs require a careful trade-off analysis. Recognizing that humans are prone to making mistakes, and that noisy feedback does not necessarily guarantee better results. In addition, integrating humans into the ML loop presents usability challenges that need to be addressed for practical reasons. In addition, challenges inherent to ML models remain, including computational complexity and typically high model latency. A primary goal of HiL systems should be to keep human effort highly rewarding and, in most cases, minimal.

To facilitate the design and deployment of HiL systems, we propose a catalog of HiL training and operation patterns that describe best practices and proven solutions. Patterns for gathering the initial training data (P1, P2, P3, and P4) aim at reducing the number of labels needed (and human involvement) to train accurate models. As labeling a sequence of machine-selected instances (P1, P2) is often perceived as exhausting, visual approaches (P3, P4) support labelers to gain additional knowledge about the data through data visualization. However, users must be able to interpret these visualizations, which may require additional skills, such as understanding semantic relationships or similarity of inputs. In addition, frequent model re-training can be costly and time-consuming. Other patterns, based on more recent decoder-based LLMs, aim to avoid human labeling altogether by generating the training data (P6) or solving the task directly (P5). However, decoder-based LLMs require extraordinary computational resources or are usually only available via black-box APIs.

A central problem with ML models is that predictions may be considered trustworthy when they are not. While adequate explanations can build trust, there is a danger of convincing humans of false correctness. Putting a human into the operational loop of an ML-based system has been shown to improve its classification performance and reliability (P8, P9, and P10). However, the question is whether human resources are available to the extent required and whether the added value is worthwhile, i.e. whether the HiL approach significantly improves reliability compared to automated predictions. One possibility is to explicitly capture human feedback in the form of corrections when working with a system (P11). Since the knowledge of trained ML models is static, frequent re-training of the models is required to maintain a certain level of reliability (P8). It should be noted that human labelers, like machines, are prone to errors and may provide incorrect labels. Collaboration mechanisms such as visualizations (P3, P4, P11), explanations (P7, P8), and uncertainty estimates (P1, P2, P10) play a crucial role in the implementation of these patterns, which are orthogonal to the HiL patterns. These mechanisms enhance the overall understanding and effectiveness of the HiL approach.

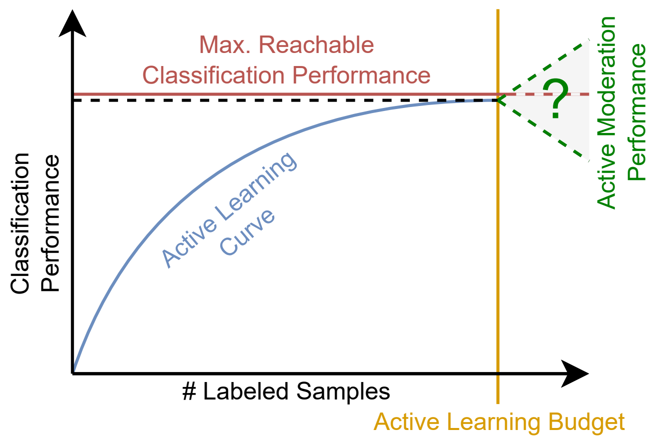


Figure 4.21: A combination of the Active Learning and Active Moderation patterns.

The described patterns can and certainly should be combined. For instance, as depicted in Figure 4.21 Active Learning (P1) and Active Moderation (P10) can be combined to optimize the learning and deployment behavior with minimal human resources. During training, Active Learning is used to incrementally increase the classification performance (blue line) of the classifier. Active Moderation can then be applied during deployment to further increase the classification performance. Furthermore, a combination of Active Learning (P1) and Visual Interactive Labeling (P3) within a labeling system can leverage automated and human data selection capabilities. The feedback stream collected through operation patterns such as Recommendation System (P8), Safeguards (P9), Active Moderation (P10), and Thumbs Up or Down (P11) can be utilized for continuous model refinement (P12). In addition, the automatic generation of additional training data (P6) can be used to address identified weak spots (P7). The key is to tailor the combination of patterns to the specific requirements and challenges of the actual ML system.

Further, it is highly promising that the HiL patterns are supported by collaboration mechanisms. These are vital for some patterns, such as visualizations for Visual Interactive Labeling (P3) and Semi-supervised Visual Interactive Labeling (P4), explanations for Recommendation Systems (P8) and Trick-the-model (P7), or uncertainties for Active Moderation (P10). Especially explanations and uncertainty values can be applied in a broader sense. For instance, explanations are highly promising in facilitating human labeling tasks. Especially highlighting why a model might be false promotes human awareness of model difficulties and misconceptions, which is the target of many HiL patterns such as Trick The Model (P7), Recommendation System (P8), Active Moderation (P10), and Thumbs Up or Down (P11). Knowing what a model does not know and when it does not know is critical to securing ML models, building user trust, and engage humans in participation.

The proposed catalog is not intended to be complete. First, our pattern catalog is the result of a literature review that does not strictly meet the requirements of being substantially systematic. Only two databases were searched and the 100 most relevant and publicly available records from each were considered. Furthermore, there is no guarantee that repeating the literature search would yield the same results. It is also possible that we have missed some approaches in the literature. The research area of HiL uses many terminologies and is difficult to capture in a single search query. Second, our pattern catalog is influenced by our own experience, which adds subjectivity. Further work should extend the catalog and evaluate its suitability and usefulness in practice.

4.7 Conclusion

Deployed ML systems are imperfect to some extent, lacking reliability and accuracy. They may not be able to solve difficult tasks. Supporting ML-based text classifiers with a continuous stream of human feedback is a promising approach to counteract the challenges of purely automated approaches. This chapter has reviewed the HiL approach and highlighted its potential to improve the applicability of text classifiers in real-world domains. Due to a lack of a widely accepted definition, we refer to HiL as "*a generic semi-automated computational paradigm in which ML models and humans interact, adapt, or learn from each other to improve the applicability of ML systems*". Human efficiency is outlined as an important goal of HiL, since manual effort is typically the bottleneck of interactive systems. The current literature on HiL has been summarized and we have identified typical enablers, namely visualization, explanation, and uncertainty estimation. Due to the observed lack of design knowledge for developing such HiL systems, the chapter has proposed a catalog of 12 patterns for designing ML-based systems with HiL. Our catalog includes seven training and five operation patterns to guide developers in choosing and adapting the HiL approach. The main observations are briefly summarized:

- The existing literature on HiL is highly convoluted, and there is no common agreement on its scope or a generally accepted definition.
- The HiL approach is not limited to certain aspects of the ML process, such as training, but can also be applied during the operation of ML models.
- While many papers show the success of HiL. Critical applicability factors such as computational latency, runtime behavior, and human error are often neglected.
- HiL is highly dependent on humans. In situations where human expertise is limited or not accessible, the benefits of HiL systems may be constrained.
- Most previous HiL approaches focus primarily on the cost-effective training of ML models. However, many weaknesses of ML models only become apparent in productive use when the model is left to work on its own. HiL has much more diverse use-cases that are underrepresented in the literature.
- There are only a few tools that facilitate text classification with HiL.

Chapter 5

Computational-aware Active Moderation

Publication. This chapter is partially based on the 2022 paper "Towards More Reliable Text Classification on Edge Devices via a Human-in-the-Loop" [16], to which I contributed by developing and conducting the ML experiments, analyzing and discussing the findings, and leading the writing. In addition, parts of this chapter rely on the 2022 paper "More Sustainable Text Classification via Uncertainty Sampling and a Human-in-the-Loop" [14], for which I undertook the development and implementation of the ML experiments, analyzed and interpreted the results, and led the writing.

Contribution. The computational efficiency of text classifiers is crucial from both an environmental and financial perspective, influencing their applicability in real-world domains. This chapter proposes a novel computational-aware Active Moderation framework to improve the applicability of ML-based text classification in computationally constrained infrastructures. The proposed framework aims to make text classification much more competitive in settings where state-of-the-art classifiers are not applicable or desirable due to memory or latency constraints (low-end infrastructure) and environmental considerations. Computational efficiency is achieved by relying on lightweight classifiers and improving their classification performance by asking humans for help with uncertain predictions during operation (Active Moderation pattern). It is demonstrated that the careful selection and delegation of highly uncertain predictions – which are likely to be incorrect – to human moderators can significantly enhance the classification performance of lightweight classifiers, even exceeding that of state-of-the-art models. This chapter discusses the computational-aware Active Moderation framework to text classification and offers several qualitative analyses of its potential. We found that computational-aware Active Moderation significantly improves the classification performance compared to a purely automated approach. For five out of six datasets in our study, Active Moderation increased the F1 score from approximately 78% to at least 95%, with less than 29.6% of the data requiring manual validation. Our Active Moderation framework remained computationally efficient even on low-end infrastructure.

5.1 Motivation

There is a growing disparity in utilizing state-of-the-art ML models between laboratory and product environments [282]. The significant computational requirements imposed by recent state-of-the-art ML models are a primary factor driving this gap. Text classifiers typically trade off classification performance with complexity, resulting in significant computational efforts required for training and inference. Highly complex deep NNs, such as LLMs, have received considerable research attention due to their potential for superior classification performance. Benchmark studies investigating such complex models, which continue to grow in size, are typically conducted in high-end laboratory settings [88, 371]. For example, the latest GPT-4 model contains 1.76 trillion parameters, rendering its adaptability to low-end infrastructure with limited main memory infeasible. By comparison, BERT already contains 110 million parameters. The high computational demands of state-of-the-art classifiers raise two critical concerns.

First, real-world deployment environments are typically much less advanced and suffer from resource constraints. State-of-the-art classifiers may be prohibitively expensive to deploy on low-end computing infrastructure [294], thus excluding practitioners from using them. For example, fine-tuning BERT requires a GPU with at least 12 GB of main memory. As a workaround, simpler and less accurate models, such as traditional ML, are often used to maintain deployment in computationally constrained (low-end) infrastructure. However, traditional ML, or even more traditional non-LLM-based deep learning models, are generally expected to deliver much worse classification performance. For example, Corazza et al. [73] report an F1 score of 82% for detecting hate speech in online forums using a traditional word embedding-based classifier, which may not meet the needs of forum providers. Other tasks, such as classifying software requirements, are even more difficult. For example, Silva-Rodríguez et al. [342] report an F1 score of only 49% for their best-performing classifier. This raises the question of how to achieve a certain level of classification performance on a low-end infrastructure that still meets the requirements of the application domain. This necessitates the consideration of a trade-off between computational capabilities and classification performance [157].

Second, the high computational requirements of recent models raise environmental concerns [6, 205, 353]. For instance, training the default BERT model can take more than three days, even when using a GPU, generating a significant amount of carbon emissions, comparable to the emissions of a transatlantic flight [353]. The question is whether such high computational efforts justify the additional classification performance of state-of-the-art classifiers or whether weaker ML models are already good enough. *Green Learning* [205] is an emerging ML paradigm characterized by low carbon footprints, lightweight models, and low computational complexity. This new focus has led to the development of the *Climate Protection Foundation*[1], which pursues the vision of reducing greenhouse gas emissions caused by software. Among other suggestions, they provide a Green Software Pattern [284] that promotes the use of energy-efficient ML models with similar functionality. A key objective of Green Learning is to maintain the classification performance of a model while significantly reducing its computational cost [6]. Environmental awareness has led to a growing reluctance computationally expensive models that consume orders of magnitude more energy for a small or negligible contribution to the actual classification performance. We use the term "**computational-aware**" to refer to the lightweight nature of an ML model that requires little computational resources.

When highly complex text classifiers are not applicable, feasible, or desirable, practitioners must resort to a less computationally intensive approach. **Active Moderation** (Section 4.5.4) is a promising HiL operational pattern to improve the classification performance of a model during deployment. It is concerned with classifiers that provide predictions for their inputs and a degree of certainty about the outcome. Improving classification performance comes at the cost of human intervention. However, human involvement must be limited to a fallback option, as manual effort can generally not be excessively spent. It is essential to let humans only intervene when the model fails to make reliable decisions. Improving the classification performance of lightweight classifiers by including a human in the ML loop is a promising approach to make text classifiers much more applicable to real-world use-cases while promoting Green Learning.

5.2 Conceptual Framework

This section describes our computational-aware Active Moderation framework for text classification. It outlines the problem statement and describes how our framework can be effectively implemented on low-end infrastructure.

[1] `https://greensoftware.foundation`

5.2.1 Problem Statement

While a properly trained classifier (traditional or deep learning) is capable of solving the majority of cases, there exist a few cases, typically less than ~ 20% [92, 217, 302, 304, 341], where a classification model fails to provide the correct class labels. We propose a computational-aware Active Moderation framework to prevent low confidence and likely incorrect classification results in low-end production infrastructure. The overall goal is to drastically improve the classification performance and thus the applicability of text classification systems during deployment.

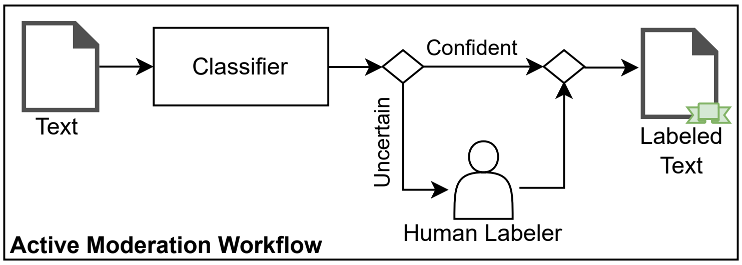


Figure 5.1: The Active Moderation workflow.

Figure 5.1 illustrates the process of Active Moderation. Initially, text instances are classified by an already trained classifier during deployment. Along with the classification outcome, the uncertainty of the prediction is estimated. Uncertainty estimates serve as an indicator of confidence and reliability (Section 2.2.1). Intuitively, uncertain instances should be assigned to a human for verification, while a confident prediction can be considered correct.

We formalized the process of Active Moderation as follows: given an already trained classification model f^{ω} with its optimized parameters ω. First, each input x_i of an unseen data corpus X is fed into the model, and a class label $y_i = f^{\omega}(x_i) \in Y$ is computed. Second, for each predicted label y_i, an uncertainty metric u_i is determined. A misclassification occurs when the predicted label y_i of an input x_i does not match the underlying ground truth $\hat{y}_i$, thus $y_i \neq \hat{y}_i$. To evaluate misclassifications, we use a binary error variable $e_i := \mathbb{I}[y_i \neq \hat{y}_i]$, which indicates whether the classification is correct. The error variable e_i takes the value 1 if $y_i \neq \hat{y}_i$, and 0 otherwise. Active Moderation aims to reduce errors by assigning error-prone instances to human labelers.

The effectiveness of Active Moderation comes from focusing human effort on instances where the model is likely to make classification mistakes. Clearly, human effort is wasted on reviewing instances that the model would have decided correctly on its own. Therefore, humans should only be involved in the classification process when their assistance is highly beneficial. Active Moderation uses the prediction uncertainty (Section 2.2.1) to determine whether an instance should be classified manually or automatically. It is assumed that the more uncertain a prediction is, the more likely a manual moderation is required. Formally, Active Moderation attempts to partition X into two discrete sets X_H and X_A, with $X_H \cup X_A = X$ and $X_H \cap X_A = \emptyset$. The set $X_H \subseteq X$ represents all elements that are moderated by humans. Therefore, X_H should contain the most uncertain and probably misclassified instances. On the other hand, $X_A \subseteq X$ describes the set of instances to be classified automatically. The model should classify these instances very accurately, and they should not require manual correction. A separation criterion is sought to determine whether a new element $x \in X$ to be classified should belong to X_A or X_H when the classifier is used in production.

We consider the prediction uncertainty and a fixed moderation effort k as the separation criteria. To enable the most effective Active Moderation process using prediction uncertainty, the 3-tuples of inputs, predicted class labels, and uncertainties $r_i := \left(x_i, y_i, u_i\right)$ are arranged in an ordered sequence $r := \langle r_i | i \in \{1, \ldots, n\}\rangle$ that best satisfies the following conditions:

1. The predicted instances are sorted by descending uncertainty, i.e., r_i comes before r_j if and only if $u_i \geq u_j$.
2. The first k instances of the sequence r should have the highest misclassification rate. The sum of their error variables should be maximal.
3. The last $n - k$ instances of the sequence r should have the lowest misclassification rate. The sum of their error variables should be minimal.

Manually annotating the instances in r according to their uncertainty is most rewarding in terms of correcting misclassifications. This leads to the following maximization problem for the task of Active Moderation:

$$\text{maximize} \sum_{i=1}^{k} e^{(i)} - \sum_{j=k+1}^{n} e^{(j)} \tag{5.1}$$

where $e^{(i)}$ is the error variable of the ith element of the sequence r. Alternatively, other performance measures can be optimized accordingly, such as maximizing the F1 score of both manually and automatically classified instances.

The variable k determines the number of instances to be moderated by a human and is initially set according to the feasible amount of supervision. In our computational-aware Active Moderation framework, the first k instances, denoted by the set $X_H = \{r^{(i)} | 1 \le i \le k\}$ undergo manual review, while the remaining set $X_M = \{r^{(i)} | k+1 \le i \le n\}$ is classified automatically. A human oracle $o_H : X \to Y$ performs the classification for the manual review. Understanding how Active Moderation performs in low-end infrastructures under varying levels of human noise is crucial for designing applicable and effective HiL classification systems that can be deployed in diverse settings, including those with limited computational resources.

5.2.2 Lightweight Text Classifiers

While previous studies on text classifiers have mainly focused on maximizing the classification performance of fully automated approaches, our goal is to evaluate the suitability of text classifiers in the context of Active Moderation. We focus on traditional ML models and shallow NNs to be compatible with low-end infrastructure. It remains uncertain which model is most efficient and how well it performs when humans are involved in the classification process under tight computational constraints.

We investigate the lightweight ML classifiers described in Section 2.1.6 for computational-aware Active Moderation. The following section briefly outlines how the conditional class probabilities can be obtained for each classifier to allow for uncertainty assessment.

Decision Tree (DT). DTs can estimate conditional class probabilities by reporting pre-computed proportions of correct class outcomes for each leaf node during training [271].

Random Forest (RF). The conditional class probability of an RF can be derived as the fraction of trees that vote for a particular class outcome, that is,

$$p(y = c|x) = \frac{1}{|T|} \sum_{t \in T} \mathbb{I}(y_t = c) \tag{5.2}$$

where T is a set of DTs, y_t the class estimate of $t \in T$ and $\mathbb{I}(y_t = c) = 1$ if $y_t = c$ otherwise 0.

k-Nearest Neighbor (kNN). Analogous to Eq. 5.2, we consider the fraction of votes as the conditional class probability of a kNN classifier.

Gaussian Naive Bayes (GNB). NB classifiers apply Bayes' rule to directly infer conditional class probabilities, that is:

$$p(y = c|x) = p(y) \prod_{i=1}^{n} P(a_i|y) \tag{5.3}$$

Since the encodings extracted from encoder-based LLMs consist of continuous and potentially negative attributes, we apply GNB. This variant assumes that the attributes of the feature vector are distributed according to a normal distribution.

Support Vector Machine (SVM). SVMs provide distances to classification boundaries rather than class probabilities. Platt scaling [289] is adapted to convert distances into probabilities. Platt scaling is an algorithm that learns probabilities from the SVM's classification score $f(x)$, that is,

$$p(y = 1|x) = \frac{1}{1 + \exp(af(x) + b)} \tag{5.4}$$

where $a, b \in \mathbb{R}$ are scalar parameters that can be optimized using the negative log-likelihood loss over the validation set.

Logistic Regression (LR). LR directly predicts conditional class probabilities alongside the class label. The class probability is given by:

$$p(y = c|x) = \phi_{softmax}(\theta x)_c \tag{5.5}$$

where θx is the linear predictor function.

Multilayer Perceptron (MLP). An MLP derives the conditional class probability by applying a *softmax* function to the network's logits z that is:

$$p(y = c|x) = \phi_{softmax}(z)_c \tag{5.6}$$

Bayesian Multilayer Perceptron (B-MLP). We also consider a Bayesian variant of the MLP. We apply MCD to approximate the conditional class probability by averaging T Monte Carlo samples over possible weights:

$$p(y = c|x, D) \approx \frac{1}{T} \sum_{t=1}^{T} p(y = c|x, \omega_t) \tag{5.7}$$

5.2.3 Text Features

We utilize contextualized BERT and Sentence-BERT (SBERT) [308] encodings as feature representations for text. SBERT is a variant of BERT that provides semantically meaningful encodings for unlabeled text documents without requiring domain-specific pre-training and fine-tuning. In addition, SBERT encodings are more computationally efficient than BERT.

5.2.4 Uncertainty Assessment

For Active Moderation to be effective, a classifier must offer a reasonable ranking of misclassifications based on the reported uncertainty values. In our experiments, we quantify the prediction uncertainty by computing Shannon's entropy [334] of the conditional class probabilities (Eq. 2.34). A comprehensive uncertainty assessment for the B-MLP is performed by calculating Shannon's entropy on the mean conditional class probabilities (Eq. 5.7).

5.3 Study Design

This section outlines the design of our computational-aware Active Moderation study. First, it presents the research questions and outlines our benchmark criteria. It then describes the dataset used and provides implementation details for the benchmark evaluation.

5.3.1 Research Questions

We aim to answer the following research questions:

RQ1: How accurately do different lightweight classifiers estimate prediction probabilities?

We evaluate the quality of the prediction probabilities reported by the lightweight classifiers. Well-calibrated class probabilities are critical for reliably assessing the true probability of predictions. The accuracy of prediction uncertainties is likely to affect the effectiveness of Active Moderation.

RQ2: Which lightweight classifier can capture the highest proportion of misclassifications via uncertainty sampling, resulting in the highest macro F1 score after removal?

For Active Moderation to be effective, the human should be assigned as many misclassified instances as possible, while the ML model should decide for itself (no human needed) which instances it can handle with high classification performance. We examine the F1 score of different classifiers when a certain proportion of the most uncertain predictions are removed from the test dataset. In the next step, these instances would undergo manual classification.

RQ3: How much of the most uncertain classification outcomes must be manually annotated to reach a certain level of macro F1 score?

Next, our investigation evaluates the F1 score attained by the proposed computational-aware Active Moderation framework. It assesses the number of instances a human annotator needs to moderate to reach a specified F1 score. Since achieving a target F1 score is usually more desirable than allocating a fixed level of human effort, such as 10%, we adhere to a classification performance-based evaluation criterion. Human annotators may introduce noise into the labeling process. Therefore, we investigate different levels of human noise, where human annotators also make mistakes.

RQ4: What is the performance difference between BERT and SBERT encodings for computational-aware Active Moderation?

Various encoder-based LLMs offer meaningful feature representations for text instances. To determine the most effective configuration for computational-aware Active Moderation, we assess the suitability of two distinct encodings for our proposed framework: BERT and SBERT.

RQ5: How efficient are different classifiers in terms of training and inference time?

One of the main goals of our study is to enhance the applicability of text classifiers on low-end infrastructure. We analyze the time required to perform training and inference to assess the feasibility of the considered classifiers on low-end infrastructure. For practical applicability, processing times should ideally be in the order of seconds or a few minutes rather than hours.

RQ6: How much training and inference time can be saved by upgrading from 4 GB to 8 GB of main memory?

Finally, we investigate the scalability of the classifiers employed. The question is how much a doubling of the main memory, i.e., from 4 GB to 8 GB, affects the total computation time, or whether 4 GB is sufficient to maintain fast training and inference times.

5.3.2 Benchmark Criteria

To address RQ1, we evaluate the quality of the prediction probabilities provided by all nine classifiers. We calculate the Brier score [49] for each classifier applied to each dataset. The Brier score serves as an indicator of the accuracy of the conditional class probabilities. The Brier score ranges from 0 to 1, where 0 indicates perfect accuracy and 1 indicates poor accuracy. It is computed as the squared error between the predicted probabilities and the true class outcomes, that is:

$$BS = |Y|^{-1} \sum_{y \in Y} \sum_{c \in C} \left(p(y = c|x) - \mathbb{I}(\hat{y} = c)\right)^2 \tag{5.8}$$

where $\mathbb{I}(\hat{y} = c) = 1$ if the true class of x represented by $\hat{y}$ is c otherwise 0. To answer RQ2, we compare the macro F1 score of all nine classifiers when a certain number of the most uncertain data instances, in our case 0%, 10%, 20%, and 30%, are removed from the test dataset. Macro-averaging is applied because several of the datasets considered have significant imbalances between classes. The objective is to give equal weight to all classes in the evaluation. For RQ3, we quantify the manual effort required by a human to achieve a given target F1 score, which involves correcting classifier outcomes. Human effort is measured by the number of cases that a human must decide manually. In our experiments, human annotations are simulated by selecting the ground truth label for each annotation request, a common approach when evaluating interactive ML approaches [340]. Since human annotations can be noisy, we simulate four levels of human noise. Specifically, each annotation request is assigned a randomly selected class label with probabilities of 0%, 5%, 10%, or 15% instead of the ground truth label. The F1 score of a combined human-automated classifier is computed based on the unified sets of manually corrected and automatically inferred labels. Regarding RQ4, all experiments are conducted with BERT and SBERT encodings to determine the most suitable one for our framework. In the next step, the training and inference time is measured to evaluate the computational efficiency of the text classifiers (RQ5). All experiments are performed on an Intel® Xeon® Gold 5115 CPU @ 2.40GHz with one core and 4 GB of main memory, which we consider to be a low-end infrastructure. Finally, we consider the computation time of an ML model as a user-centric indicator of its computational-awareness. For RQ6, we examine the time required for training and inference when utilizing 4 GB and 8 GB of main memory. All reported measurements represent the mean of five stratified cross-fold datasets with a 50% train-test split. Throughout the remainder of this chapter, the abbreviated term “F1 score” will refer to the macro F1 score.

5.3.3 Datasets

We aim at a domain-independent evaluation of computational-aware Active Moderation. Therefore, we focus on various tasks involving real-world data from online forums, marketplaces, issue trackers, and emails. We consider six publicly available real-world datasets covering heterogeneous classification tasks. These datasets include binary and multi-class classification tasks that are originally skewed. Table 5.1 summarizes the key statistics of the datasets.

Dataset	Size	$\|C\|$	Class Distribution	#Words ($\mu \pm \sigma$)
App Store	5,752	3	3,472:1,286:994	24 ± 29
News	13,919	4	4,891:3,979:2,625:2,424	198 ± 711
Hate Speech	24,783	2	19,190:5,593	15 ± 7
Issues	29,998	3	10,000:10,000:9,998	54 ± 59
Reuters	8,614	8	3,930:2,319:527:499:458:425:290:166	117 ± 129
TREC	5,952	6	1,344:1,300:1,288:1,009:916:95	10 ± 4

Table 5.1: Dataset details, including size, number of classes, class distribution, and the mean and standard deviation of words per text instance.

The **App Store** dataset [239] encompasses user-written reviews sourced from app stores. The reviews are manually annotated as *feature request*, *bug report* or *praise*. Second, we consider the **20NewsGroups** dataset [212], which contains messages categorized into 20 different classes. The dataset is reorganized to focus on the four most frequent classes: *comp*, *politics*, *rec*, and *religion* [150]. The **Hate Speech** dataset [80] features a binary classification task to detect toxic messages (*hate speech* or *offensive language*) from Twitter. Third, the **Issues** dataset [179] consists of software issues extracted from heterogeneous GitHub projects. Issues were queried from 12,112 projects and labeled as *bug*, *enhancement* (i.e., an improvement or new feature request), or *question*. The **TREC** dataset [226] is a collection of questions, each referring to one of six different answer types: *person*, *location*, *number*, *abbreviation*, and *entity*.

5.3.4 Implementation Details

As text features, we use the pre-trained *bert-base-uncased*[2] model for BERT and the *all-mpnet-base-v2*[3] model for SBERT, both of which compute encodings of length $n = 768$. For most classifiers, we rely on the default implementation provided by the Scikit-learn library[4]. For classifiers not available in Scikit-learn, we implemented them using TensorFlow[5]. For the DT classifier, Scikit-learn's

[2] `https://huggingface.co/bert-base-uncased`
[3] `https://huggingface.co/sentence-transformers/all-mpnet-base-v2`
[4] `https://scikit-learn.org/stable/index.html`
[5] `https://www.tensorflow.org/`

non-parametric implementation is utilized. We use $T = 100$ DTs for the RF classifier and set $k = 25$ for the kNN classifier. A linear kernel is used for the SVM. The structure of the MLP is configured as [768, 500, 500, C]. No hyper-parameter tuning is performed. Since Scikit-learn does not support a Bayesian-MLP, we used Tensorflow for its implementation (B-MLP*). We approximate the posterior using MCD and apply $T = 100$ forward passes. Since replicating Scikit-learn's MLP in TensorFlow is challenging, we develop a conventional non-Bayesian MLP, denoted as MLP*, to better compare the impact of Bayesian modeling. Unlike Scikit-learn's MLP implementation, our MLP model applies dropout similarly to the Bayesian MLP, but only during training. In addition, for all MLP implementations, we designate 10% of the training data as validation data to facilitate early stopping.

5.4 Results

This section presents the results of our experiments and addresses the six research questions. Our replication package is available online[6].

5.4.1 Quality of Predicted Probabilities

First, we assess the quality of the predicted probabilities produced by each classifier. Figure 5.2 illustrates the mean class probability for both wrong (misclassified) and correct classifications utilizing the App Store dataset. Each box represents the lower and upper quartiles, while a line denotes the median. The whiskers portray the range of the data. The figure indicates evident differences in the probability distributions between misclassified and correctly classified text instances. For example, on the App Store dataset using SBERT, the mean class probability across all classifiers for all misclassifications and correct predictions is 66.4% and 89.42%, respectively. Only the DT and GNB classifiers demonstrate no disparities in their means. For most classifiers and all datasets, the reported class probabilities allow to distinguish whether the predictions are likely to be misclassified or correct. On average, BERT and SBERT show no significant differences. We obtained similar results for the other datasets.

Next, we examine the Brier scores of all nine classifiers across all six datasets, as shown in Table 5.2. The table reveals large differences among the classifiers regarding the quality of their class probabilities. On average, a DT and GNB exhibit the worst calibrated probabilities, with a Brier score of 0.70 and 0.40, respectively, using SBERT encodings. RF and kNN achieve nearly equivalent

[6] `https://github.com/jsandersen/MRTviaHIL`

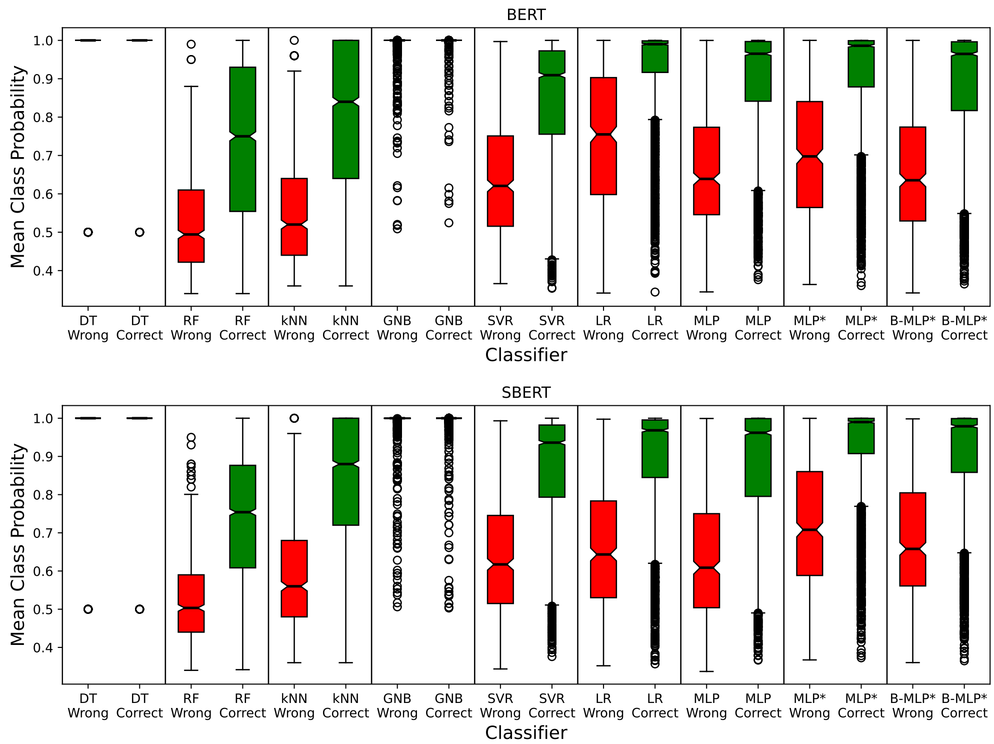


Figure 5.2: Distributions of mean class probabilities for all wrong (misclassified) and correct classification results for the App Store dataset.

calibrated probabilities, averaging above 0.27. The best calibrated probabilities, with average Brier scores below 0.25, are attained by SVM, LR, as well as MLP and its variants. Across all datasets, MLP and MLP* yield the most favorable results using BERT encodings, while LR, MLP, and MLP* exhibit superior performance using SBERT. Particularly, MLP* yields the most accurate probabilities with average Brier scores of approximately 0.22 (BERT) and 0.20 (SBERT). Similar results were observed for the other datasets.

5.4.2 Classifier Performance Under Stepwise Removal of Uncertain Instances

We examine the F1 score of the automatically classified instances of our framework. This is the subset of the test data that the ML-based text classifier has to decide on (no human involvement). Table 5.3 presents the F1 scores of the nine classifiers applied to each of the six datasets using SBERT encodings. Table A.1 illustrates the results of BERT. The columns represent the F1 scores obtained when a certain number (0%, 10%, 20%, and 30%) of the most uncertain instances are removed from the test dataset. Each cell also shows the relative

	Classifier	App Store	News	Hate Speech	Issues	Reuters	TREC	AVG
BERT	DT	0.6639	0.6983	0.5230	1.0166	0.5699	1.0181	0.7483
	RF	0.3344	0.3392	0.2549	0.5041	0.2632	0.4764	0.3620
	kNN	0.3105	0.3021	0.2717	0.4992	0.2065	0.3805	0.3284
	GNB	0.6100	0.4206	0.4605	0.7668	0.4148	0.6990	0.5620
	SVM	0.2686	0.2097	0.2118	0.4392	0.1125	0.2208	0.2438
	LR	0.2551	0.1858	0.2089	0.4320	0.1009	0.2151	0.2330
	MLP	0.2419	0.1920	0.2212	0.4385	0.1098	0.2336	0.2395
	MLP*	0.2390	0.1760	0.2046	0.4259	0.1028	0.2086	0.2262
	B-MLP*	0.2322	0.1728	0.2000	0.4209	0.1005	0.2060	0.2221
	AVG	0.3506	0.2996	0.2841	0.5492	0.2201	0.4065	0.3517

	Classifier	App Store	News	Hate Speech	Issues	Reuters	TREC	AVG
SBERT	DT	0.6534	0.6126	0.4551	1.0478	0.4182	1.0234	0.7018
	RF	0.3174	0.2779	0.2137	0.5191	0.1748	0.4931	0.3327
	kNN	0.2871	0.1292	0.2151	0.5099	0.0790	0.4437	0.2773
	GNB	0.4418	0.2286	0.3506	0.6958	0.1986	0.4929	0.4014
	SVM	0.2469	0.1546	0.1699	0.4115	0.0519	0.1872	0.2037
	LR	0.2381	0.1410	0.1671	0.4081	0.0489	0.1848	0.1980
	MLP	0.2427	0.1453	0.1822	0.4422	0.0533	0.1992	0.2108
	MLP*	0.2361	0.1377	0.1681	0.4139	0.0495	0.1792	0.1974
	B-MLP*	0.2309	0.1349	0.1629	0.4055	0.0483	0.1768	0.1932
	AVG	0.3216	0.2180	0.2316	0.5393	0.1247	0.3756	0.3018

Table 5.2: Brier scores of the different classifiers as an indicator of the accuracy of the predicted conditional class probabilities. A lower Brier score indicates a better calibration. The two best and worst Brier scores for each dataset are highlighted in green and red, respectively.

improvement in F1 score compared to the previous removal rate. For instance, an LR with SBERT on the News dataset achieves an F1 score of 88.90% on the entire test dataset. By removing 10% of the most uncertain instances from the evaluation, the F1 score increases to 93.78%, indicating a relative improvement of 5.49%.

Our experiment reveals that when classifying the whole test dataset, a DT provides the worst F1 score on average, followed by GNB, RF, and kNN. LR, SVM, MLP, MLP* and the B-MLP* achieve on average the highest initial F1 scores as well as the highest scores when a certain number of the most uncertain instances have been removed from the test dataset. For example, an improvement of 4.7% is reached when 10% of the data is removed. The kNN performs exceptionally well only on the News dataset with SBERT encodings but poorly on the other datasets. The MLP implementation of Scikit-learn performs worse than our TensorFlow implementation. Bayesian modeling (B-MLP*) does not improve the F1 score compared to a deterministic MLP. When comparing SBERT encodings to BERT, all classifiers achieve up to 20% higher F1 scores at the same removal ratio. In general, the F1 score increases substantially when a certain proportion of the most uncertain instances are removed

Classifier	App Store				News				Hate Speech			
	0%	10%	20%	30%	0%	10%	20%	30%	0%	10%	20%	30%
DT	58.48	58.58	58.47	58.40	66.25	67.28	67.21	67.15	67.64	67.59	67.72	67.56
		+0.16	-0.18	-0.13		+1.55	-0.10	-0.08		-0.07	+0.19	-0.24
RF	63.44	65.30	65.86	65.34	87.24	91.44	94.80	96.24	76.82	76.93	74.07	67.97
		+2.93	+0.86	-0.80		+4.82	+3.68	+1.51		+0.14	-3.72	-8.24
kNN	69.42	70.47	69.94	69.41	89.67	94.63	97.73	99.06	71.05	71.07	71.83	73.88
		+1.50	-0.75	-0.75		+5.53	+3.28	+1.37		+0.03	+1.07	+2.85
GNB	74.11	77.35	80.45	83.33	86.55	91.94	95.48	96.80	77.19	80.71	83.94	86.88
		+4.37	+4.01	+3.59		+6.22	+3.85	+1.39		+4.56	+4.00	+3.51
SVM	76.46	80.13	83.68	87.66	88.51	92.92	96.20	97.43	82.79	86.25	88.67	89.87
		+4.80	+4.43	+4.76		+4.98	+3.53	+1.28		+4.18	+2.81	+1.35
LR	77.29	81.41	85.34	89.27	88.90	93.78	96.92	98.11	82.90	86.78	89.51	90.88
		+5.32	+4.83	+4.60		+5.49	+3.35	+1.22		+4.68	+3.15	+1.53
MLP	78.31	82.21	85.29	87.95	89.16	94.42	97.46	98.62	83.14	86.92	89.95	92.12
		+4.98	+3.74	+3.12		+5.89	+3.22	+1.19		+4.55	+3.48	+2.41
MLP*	78.75	82.99	86.43	90.44	89.28	94.64	97.60	98.85	83.71	87.63	90.96	93.02
		+5.38	+4.14	+4.65		+6.00	+3.13	+1.28		+4.68	+3.80	+2.27
B-MLP*	78.68	82.72	86.43	90.25	89.27	94.53	97.58	98.79	83.69	87.61	90.75	92.69
		+5.12	+4.50	+4.42		+5.90	+3.22	+1.24		+4.68	+3.59	+2.14
AVG	72.77	75.68	77.99	80.23	86.09	90.62	93.44	94.56	78.77	81.28	83.04	83.87

Classifier	Issues				Reuters				TREC			
	0%	10%	20%	30%	0%	10%	20%	30%	0%	10%	20%	30%
DT	47.38	47.36	47.38	47.41	65.12	65.42	65.43	65.22	46.02	45.81	45.72	45.42
		-0.05	+0.04	+0.07		+0.45	+0.01	-0.32		-0.45	-0.21	-0.65
RF	62.68	64.99	66.64	67.94	86.85	91.48	92.61	94.27	69.51	72.10	73.92	76.32
		+3.70	+2.54	+1.95		+5.33	+1.24	+1.79		+3.71	+2.53	+3.24
kNN	60.60	62.91	64.76	66.81	88.73	95.12	99.45	99.70	69.53	72.03	74.07	76.37
		+3.81	+2.96	+3.16		+7.20	+4.55	+0.25		+3.59	+2.83	+3.11
GNB	61.79	63.44	64.97	66.53	88.49	95.02	98.44	99.47	74.33	77.52	80.45	82.76
		+2.67	+2.41	+2.39		+7.38	+3.60	+1.05		+4.30	+3.77	+2.88
SVM	70.32	73.08	75.28	77.61	93.85	98.19	99.65	99.92	86.21	89.17	91.72	93.84
		+3.92	+3.01	+3.11		+4.62	+1.49	+0.28		+3.43	+2.86	+2.31
LR	70.45	73.24	75.68	78.04	93.93	99.05	99.84	99.84	86.11	89.12	91.64	93.75
		+3.96	+3.33	+3.12		+5.45	+0.80	-0.00		+3.49	+2.82	+2.31
MLP	69.47	72.33	74.86	77.33	93.33	98.52	99.66	99.62	86.40	90.33	93.14	94.36
		+4.11	+3.50	+3.30		+5.57	+1.16	-0.04		+4.55	+3.11	+1.31
MLP*	70.99	73.93	76.47	79.12	93.96	99.15	99.81	99.90	86.51	89.79	92.14	92.46
		+4.14	+3.44	+3.47		+5.53	+0.67	+0.09		+3.78	+2.62	+0.34
B-MLP*	71.01	73.87	76.31	78.72	94.03	99.22	99.80	99.92	86.45	89.41	91.77	91.53
		+4.03	+3.30	+3.16		+5.52	+0.58	+0.12		+3.42	+2.63	-0.26
AVG	64.97	67.24	69.15	71.06	88.70	93.46	94.97	95.32	76.79	79.48	81.62	82.98

Table 5.3: F1 scores of different classifiers after removing a certain proportion of the most uncertain predictions from the test dataset, employing SBERT embeddings. The table also shows the relative changes from the previously removed portion. The two best and worst F1 scores for each setting are highlighted in green and red, respectively.

from the test dataset. Only the uncertainty estimates of a DT are consistently unable to detect misclassifications. In this case, the F1 score does not improve with larger removal ratios. In particular, the relative F1 score improvements decrease with larger removal ratios. On average, classifiers with high initial F1 scores achieve the best F1 scores after removing uncertain instances.

	App Store				**News**				**Hate Speech**				
Classifier	89%	91%	93%	95%	91%	93%	95%	97%	89%	91%	93%	95%	
DT	73.12	78.04	82.76	87.73	72.70	78.98	84.84	90.88	66.28	72.66	78.70	84.80	0% human noise
RF	32.45	36.83	41.36	46.75	6.48	10.70	15.10	23.82	18.30	22.68	28.08	35.31	
kNN	27.35	31.35	36.23	41.96	2.07	5.30	9.43	15.39	19.70	23.14	27.65	34.28	
GNB	31.85	36.83	42.77	50.94	6.65	10.13	14.32	22.47	25.83	31.65	38.88	48.95	
SVM	19.31	23.37	27.82	33.26	4.34	8.58	12.76	19.21	9.72	14.01	19.21	26.76	
LR	17.55	21.21	25.69	31.04	3.15	6.68	10.95	16.51	9.28	13.24	18.19	25.00	
MLP	17.18	21.18	25.66	31.91	2.84	6.06	9.67	14.97	9.37	13.77	18.88	26.57	
MLP*	15.93	20.15	24.47	29.32	2.57	5.62	9.48	14.45	8.17	11.93	16.80	23.50	
B-MLP*	16.27	20.24	24.53	29.60	2.57	5.63	9.58	14.57	8.17	11.96	16.74	23.48	
DT	81.04	86.23	91.49	96.87	81.97	88.81	95.80	-	73.77	80.76	87.65	94.81	5% human noise
RF	36.14	41.36	47.43	57.01	7.04	11.61	17.86	36.09	20.01	25.36	32.27	44.16	
kNN	30.35	35.61	42.18	51.60	2.23	5.65	10.53	17.57	21.22	25.31	31.21	42.68	
GNB	35.36	41.86	51.19	64.08	7.08	10.93	16.16	40.73	28.31	35.95	46.34	66.19	
SVM	21.34	25.88	31.70	40.86	4.73	9.18	14.38	25.46	10.63	15.29	21.51	34.64	
LR	19.18	23.15	28.94	38.17	3.33	7.28	12.33	20.11	9.89	14.44	20.41	30.39	
MLP	18.99	23.53	30.38	38.45	3.05	6.51	10.80	17.70	10.17	15.08	21.76	33.45	
MLP*	17.30	22.06	27.03	35.70	2.76	5.98	10.33	16.91	8.72	12.96	18.67	28.49	
B-MLP*	17.58	22.12	27.16	35.67	2.74	6.02	10.66	17.37	8.87	13.17	18.82	28.63	
DT	91.21	97.50	-	-	95.75	-	-	-	82.63	90.86	98.65	-	10% human noise
RF	40.71	47.37	60.54	-	7.87	12.60	24.05	-	22.12	28.63	39.39	-	
kNN	34.89	42.15	54.47	-	2.43	6.18	12.40	27.46	22.87	28.16	38.48	-	
GNB	40.05	50.91	-	-	7.59	12.03	20.79	-	31.34	41.41	67.41	-	
SVM	23.44	29.22	37.83	-	5.55	9.99	16.91	-	11.31	16.89	25.43	-	
LR	21.09	26.69	34.04	-	3.61	8.07	13.56	-	10.63	15.90	24.02	-	
MLP	21.43	27.28	36.36	-	3.20	6.95	12.07	-	10.97	16.53	25.57	-	
MLP*	19.74	25.03	33.35	-	2.93	6.47	11.44	26.09	9.38	14.24	21.84	43.32	
B-MLP*	19.90	25.38	32.98	-	2.90	6.57	11.65	-	9.44	14.25	21.44	47.59	
DT	-	-	-	-	-	-	-	-	96.03	-	-	-	15% human noise
RF	48.19	-	-	-	8.56	13.84	-	-	25.39	35.64	-	-	
kNN	40.64	59.92	-	-	2.53	6.59	13.71	-	25.64	34.47	-	-	
GNB	52.35	-	-	-	8.07	12.79	-	-	37.87	-	-	-	
SVM	26.53	36.36	-	-	5.98	11.15	-	-	12.65	19.84	-	-	
LR	23.15	31.91	-	-	3.72	8.69	15.86	-	11.82	18.48	-	-	
MLP	23.90	33.67	-	-	3.43	7.61	13.89	-	12.18	19.06	-	-	
MLP*	21.68	28.44	-	-	3.07	6.98	12.80	-	10.19	16.16	29.11	-	
B-MLP*	22.22	29.13	-	-	3.03	7.04	12.73	-	10.15	16.06	30.04	-	

Table 5.4: F1 scores of computational-aware Active Moderation with SBERT encodings on the App Store, News, and Hate Speech datasets. Each cell denotes the percentage of the most uncertain classification results that require manual annotation to attain a specified F1 score at a particular level of human noise. F1 scores that are unattainable due to high levels of human misclassification are denoted by "-". The two best and worst F1 scores for each setting are highlighted in green and red, respectively.

5.4.3 Semi-automated Classification Performance

In the next step, we investigate the overall classification performance of computational-aware Active Moderation. This includes letting human annotators handle a certain amount of classification. Tables 5.4 and 5.5 display how many of the most uncertain instances from the unseen test set need to be manually classi-

Classifier	Issues 79%	81%	83%	85%	Reuters 93%	95%	97%	99%	TREC 89%	91%	93%	95%	
DT	60.17	63.94	67.76	71.60	80.33	86.32	91.57	97.33	80.11	83.23	87.37	90.76	0% human noise
RF	32.36	37.26	42.14	47.32	4.11	7.38	14.67	30.60	35.79	42.24	49.73	58.33	
kNN	34.28	38.99	43.90	48.54	3.13	5.46	9.12	14.14	43.21	49.97	56.12	64.58	
GNB	31.42	35.52	39.96	44.73	5.78	8.80	13.12	21.73	34.38	41.60	50.60	59.78	
SVM	17.14	21.94	26.96	31.73	0	1.16	4.46	11.79	5.95	10.89	16.50	24.63	
LR	16.68	21.16	26.02	31.00	0	1.23	4.36	8.87	6.22	10.99	16.73	24.36	
MLP	17.94	22.41	27.05	31.98	0	1.74	4.50	9.68	3.93	8.43	13.21	19.76	
MLP*	15.70	20.01	24.47	29.13	0	1.16	3.41	7.89	4.47	8.33	13.41	19.22	
B-MLP*	15.84	20.31	24.84	29.76	0	1.14	3.65	8.06	4.70	8.64	13.68	20.23	

DT	64.26	68.38	72.50	76.50	-	-	-	-	91.33	95.63	99.87	-	5% human noise
RF	35.21	40.61	46.21	52.09	5.13	11.93	-	-	49.53	61.06	85.28	-	
kNN	37.17	42.51	47.59	54.22	3.46	6.57	12.63	-	55.75	68.38	85.72	-	
GNB	33.76	38.34	43.52	48.67	6.59	10.45	17.04	-	47.78	60.22	81.99	-	
SVM	18.71	24.01	29.36	34.78	0	1.30	5.83	-	7.90	14.65	24.73	-	
LR	18.09	23.05	28.48	33.97	0	1.35	5.36	-	8.43	15.12	24.87	-	
MLP	19.41	24.31	29.58	34.95	0	1.93	5.46	-	5.21	10.99	17.57	42.78	
MLP*	16.97	21.61	26.47	31.08	0	1.32	4.02	-	5.95	11.29	18.85	-	
B-MLP*	17.16	21.92	26.99	32.44	0	1.30	4.20	-	6.72	12.33	18.72	-	

DT	68.92	73.28	77.59	81.85	-	-	-	-	-	-	-	-	10% human noise
RF	38.80	44.70	50.95	57.67	6.99	-	-	-	-	-	-	-	
kNN	40.98	46.47	53.24	59.50	3.71	8.24	-	-	-	-	-	-	
GNB	36.47	41.68	47.26	53.62	7.73	13.58	-	-	-	-	-	-	
SVM	20.36	26.28	32.18	38.45	0	1.37	-	-	8.53	18.85	-	-	
LR	19.60	25.25	31.22	37.46	0	1.67	7.31	-	9.14	20.53	-	-	
MLP	21.10	26.60	32.46	39.01	0	2.25	7.55	-	5.21	10.99	-	-	
MLP*	18.34	23.39	28.97	35.02	0	1.44	5.08	-	5.98	12.03	-	-	
B-MLP*	18.67	23.85	29.72	36.06	0	1.49	5.15	-	7.29	13.71	-	-	

DT	74.30	78.89	83.39	87.89	-	-	-	-	-	-	-	-	15% human noise
RF	43.10	49.88	57.83	66.28	-	-	-	-	-	-	-	-	
kNN	45.08	52.38	59.78	68.60	5.85	-	-	-	-	-	-	-	
GNB	40.11	46.35	53.38	61.50	10.45	-	-	-	-	-	-	-	
SVM	22.35	28.88	35.54	43.41	0	2.11	-	-	15.73	-	-	-	
LR	21.37	27.87	34.58	41.86	0	2.30	-	-	15.66	-	-	-	
MLP	22.97	29.20	35.76	43.54	0	3.37	-	-	8.43	-	-	-	
MLP*	19.93	25.50	31.83	39.10	0	1.65	-	-	10.48	-	-	-	
B-MLP*	20.28	26.11	32.76	39.96	0	1.97	-	-	11.93	-	-	-	

Table 5.5: F1 scores of computational-aware Active Moderation with SBERT encodings on the Issues, Reuters, and TREC datasets. Each cell denotes the percentage of the most uncertain classification results that require manual annotation to attain a specified F1 score at a particular level of human noise. F1 scores that are unattainable due to high levels of human misclassification are denoted by "-". The two best and worst F1 scores for each setting are highlighted in green and red, respectively.

fied to raise the semi-automated classification results to a given F1 score using SBERT. The results for BERT are depicted in Tables A.2 and A.3. Each subtable represents a different level of human noise. For example, on the News dataset 12.76% of the most uncertain predictions must be manually corrected to enhance the F1 score of the SVM (from the original 88.51%) to 95% using SBERT encodings.

An accurate human annotator can achieve a high F1 score of 95% to 99% by manually labeling less than 23.5% of the News, Hate Speech, Reuters, and TREC datasets. The App Store dataset requires 29.32%. The Issues dataset has the worst F1 score using purely automated classification. Here, an F1 score of 83% is achieved when 24.47% of the data is manually annotated, which is an absolute F1 score improvement of 10.01%. The tables further demonstrate that ML models with a high initial F1 scores require less manual effort to bring the F1 score to a given target level. Overall, ML models with lower initial F1 scores rarely surpass the more accurate classifiers in terms of final F1 scores. Human annotators with higher noise levels require more manual effort to achieve a certain F1 score due to the increased number of misclassifications. However, our results suggest that computational-aware Active Moderation can lead to a higher F1 score than pure machine or human efforts alone. For instance, an LR classifier with an initial F1 score of 88.90% on the News dataset and a 10% noisy human can attain an F1 score of less than 95% (maximum 96.47%) if less than 13.56% of the dataset is manually classified.

It can be seen that Active Moderation with a lightweight classifier can achieve strong accuracies even when the annotator makes multiple classification errors. In our experiments, the best-performing classifiers achieve an F1 score of 89% (+10.92), 95% (+8.84), 91% (+11.81), 79% (+9.73), 99% (+11.52), and 91% (+4.61) with a manual effort of 19.74%, 11.44%, 14.24%, 18.34%, 5.08%, and 10.99%, respectively, considering a human noise level of 10% and SBERT. Compared to a 100% accurate human annotator, this is an increase in manual effort of 19.47%, 17.13%, 16.22%, 14.39%, 32.87%, and 23.29% respectively. Our results indicate that very high F1 scores (around 95% to 99%) are not always feasible in all computational-aware Active Moderation settings. When human annotations contain significant noise, the F1 score decreases after a certain level of support. As less uncertain predictions are annotated more frequently by humans in larger workloads, classification performance decreases. This phenomenon arises because noisy humans incorrectly annotate instances that the machine would have decided correctly. SBERT encodings consistently yield higher F1 scores than BERT encodings across all noise levels. For instance, on the Hate Speech dataset, SBERT encodings require on average 9.6%, 12.0%, 13.5%, and 14.0% less manual effort to attain F1 scores of 89%, 91%, 93%, and 95%, respectively. Furthermore, higher F1 scores can be achieved with SBERT encodings that would be unattainable with BERT encodings. For instance, an F1 score of 97% would not be achievable by a 15% noisy human on the News dataset using BERT, whereas SBERT requires a manual effort of 12.8%. Overall, the SBERT

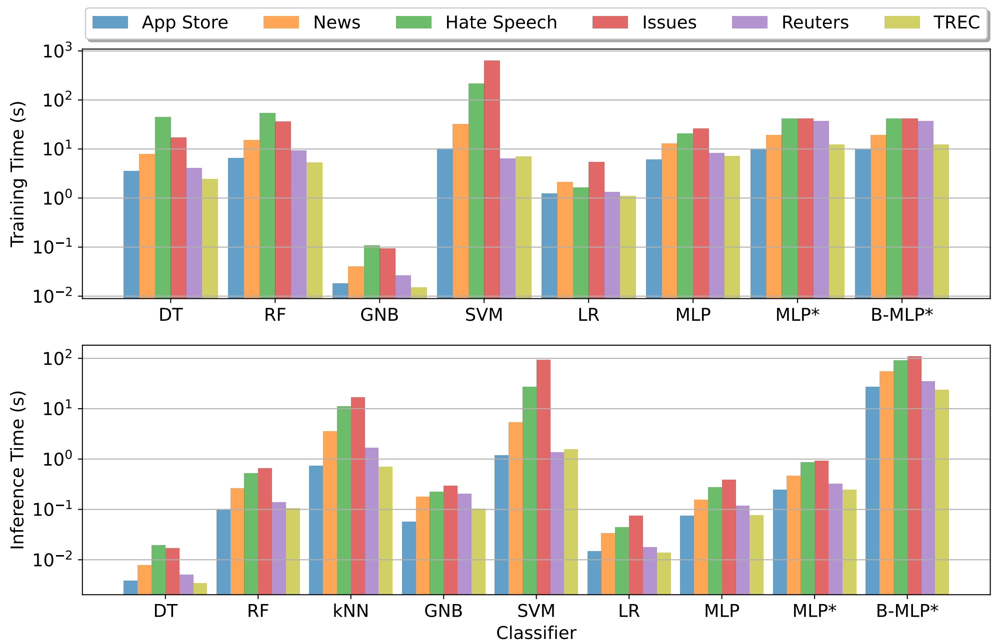


Figure 5.3: Total training and inference time of all classifiers using SBERT.

encodings reduce manual effort by ~ 5% to ~ 20%. However, the differences between BERT and SBERT were small for the App Store dataset, about 1%.

5.4.4 Runtime Investigation and Scalability

Figure 5.3 shows the training time of the classifiers using SBERT encodings. The classifiers are listed on the x-axis, and the average training time in seconds is shown on the y-axis on a logarithmic scale. The runtimes with BERT encodings are not reported as they cannot match SBERT in terms of F1 score and provide less insight. For MLP, MLP*, B-MLP*, and SVM, the training time increases by up to 53% when using BERT compared to SBERT. However, the differences are not significant when using DT, RF, and NB. kNN is not included because it uses a memory-based learning algorithm that requires no training.

A GNB classifier has the fastest training time, averaging about 0.1 seconds on 14,999 instances (Issues dataset). The LR has the second shortest training time with 5.5 seconds. A DT, MLP, and RF are much slower, with 17.21, 26.08, and 36.78 seconds, respectively. The dropout-based MLP* implementation took 40.54 seconds, almost 1.5 times as long as the MLP. MLP* and B-MLP* take the same amount of time to train because they utilize the same training procedure. The SVM is the only classifier that shows an exponential increase in training time concerning the size of the training data, ranging from 10.38 seconds for the App Store (size 2,876) to 10.67 minutes for the Issues (size 14,999) dataset.

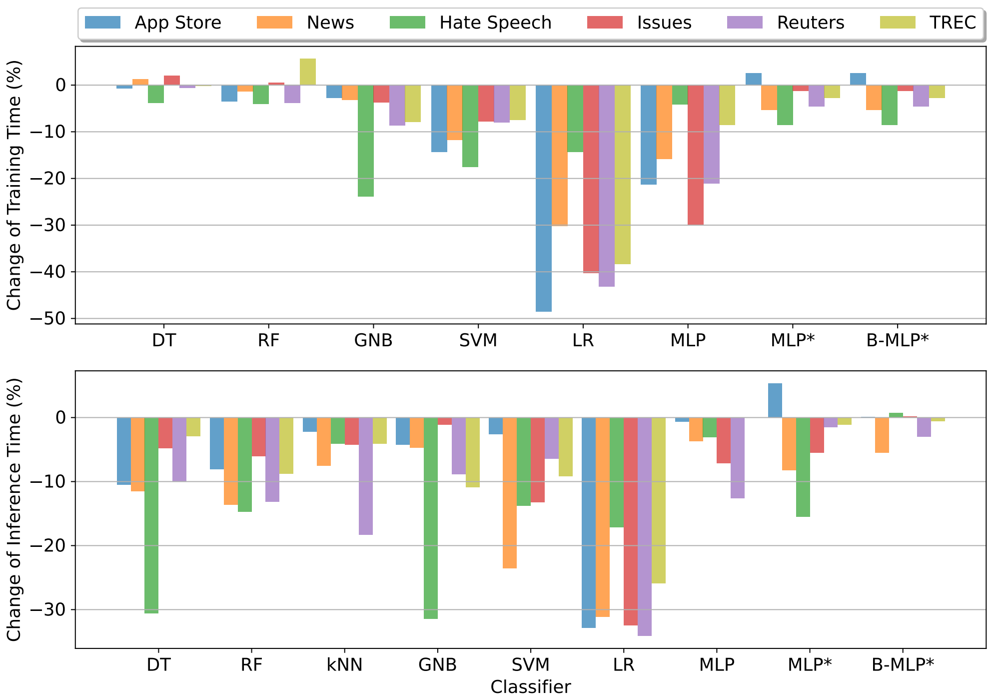


Figure 5.4: Relative time change when using 8 GB of main memory compared to 4 GB using SBERT.

Thus, SBERT/BERT-based LR classifiers are time-saving and can be trained in a few seconds while obtaining a high F1 score.

We also assess the time required for inference. The DT, RF, GNB, LR, MLP, and MLP* classifiers take less than one second to infer the labels for 14,999 instances (Issues dataset). The kNN classifier is much slower, with an inference time of 16.69 seconds. The inference time of an SVM and kNN grows exponentially regarding the number of predicted texts. The SVM needs 1.20 seconds for the App Store and 93.85 seconds for the Issues dataset. The kNN classifier needs 0.73 seconds for the App Store and 17.00 seconds for the Issues dataset. Sampling-based Bayesian approximations require more time for inference because they require multiple forward passes to approximate the class probabilities. By performing 100 forward passes, a Bayesian MLP takes 108.71 seconds (Issues dataset).

Figure 5.4 illustrates the relative time savings in training and inference when using 8 GB of main memory compared to 4 GB. The LR, SVM, and MLP benefit the most from the additional memory. For instance, on the Issues dataset, the training time can be reduced by 7.8%, 40.3%, and 29.9% for SVM, LR, and MLP, respectively, resulting in absolute time savings of 50.0, 2.22, and 7.8 seconds. For the DT, RF, GNB, MLP, and B-MLP*, the time reduction is less than 2%.

Regarding inference, the time saved is less than 1 second for all datasets except the SVM, where 12 seconds were saved. Since the training and inference times of many classifiers are very short, the absolute time savings were generally small for most classifiers. Using 8 GB instead of 4 GB memory may only be necessary for very large datasets or time-critical real-time applications.

5.5 Discussion

We discuss the implications of our findings, consider their applicability across domains, and outline the limitations of our study. Further, we review related work.

5.5.1 Implications

We evaluated the suitability of the Active Moderation pattern for improving the applicability of lightweight classifiers during their operation in computationally constrained environments. Our findings suggest that Active Moderation can effectively maintain strong classification performance while significantly reducing the use of computational resources, thereby promoting Green Learning. Our framework requires human effort, which can be costly and may not be available in unlimited amounts. Ultimately, users of HiL systems must decide whether they want to invest in human effort, address computational constraints, or reduce their carbon footprint.

We found that uncertainty estimates from lightweight classifiers are useful for detecting misclassifications. An essential aspect of the Active Moderation pattern. Although most classifiers did not use Bayesian modeling, they still provided similarly well-calibrated class probabilities as a Bayesian MLP. In particular, we found that the accuracy of the predicted class probabilities depended not only on the classifier, but also on the data being classified. For instance, an LR classifier achieves a Brier Score of 0.10 on the Reuters dataset and 0.42 on the Issue dataset. Thus, Active Moderation is expected to be much more effective in some classification settings than in others. Since classifiers with the highest F1 scores also report the best Brier scores, it is unnecessary to use one classifier for uncertainty estimation and another classifier for the classification decision. In particular, we found that computational-aware Active Moderation increased the F1 score from around 78% to at least 95% on five out of six datasets by manually validating less than 29.6% of the data using SBERT. On the Issues dataset, which initially had a relatively low F1 score of 71%, F1 scores of 83% and 85% were achieved when 24.47% and 29.13% of the data, respectively, were manually annotated. This represents an absolute F1 score improvement of 12.01% and 14.01%, respectively. In our recent paper [16], we even showed that our framework is capable of outperforming a state-of-the-art BERT model with a large margin when moderating only 12.5% of the data.

Furthermore, our results imply effectiveness even when humans are noisy and mislabel in multiple instances. Our experiment shows that even a 15% noisy human can lead to more accurate results than a purely ML-based classification. Thus, Active Moderation is a sufficient to improve the classification performance of a classifier during deployment, even if the human moderator commits some incorrect labels. Practitioners must evaluate their own behavior to assess how much effort is worth investing in the loop. However, our results also imply a decreasing human efficiency when large numbers of instances have to be decided manually. For instance, removing 10% of the most uncertain data instances, the F1 score of the remaining instances increased by up to 7.38%. Removing an additional 10% provides only an improvement of 3.60%. As a result, human efficiency dropped dramatically as workloads increased. Practitioners must be aware that arbitrary F1 score improvements may require significant human effort.

Our runtime experiments show that Active Moderation with a lightweight classifier is highly computationally efficient. Training and inference can be performed in seconds, even with access to only 4 GB of main memory. It requires only < 5.6 and < 3.3 seconds for training and inference on 14,998 data instances with 4 GB and 8 GB memory, respectively. Thus, Active Moderation has the potential to achieve very high levels of classification performance on infrastructures where state-of-the-art classification models would not be applicable or desirable due to environmental concerns or computational constraints. This makes strong classification performance available to a much wider audience. Our study provides guidelines to support practitioners in choosing the most efficient classifier for Active Moderation when humans are willing to re-label a portion of the classification results without relying on a strong computational infrastructure.

5.5.2 Field of Application

Active Moderation is a HiL operation pattern designed to enhance the classification performance of text classifiers during deployment. Its domain-independent structure makes it applicable across various application domains. The lightweight and computationally aware nature of our computational-aware Active Moderation framework makes it suitable for a wider audience that may not be able or willing to apply highly complex ML models.

Our findings are particularly relevant for domains where text classification tasks still have to be performed manually due to a lack of well-performing ML models. A promising field of applications is supporting text labeling tasks that domain experts carry out during their daily work [182, 193]. In such cases, computational-aware Active Moderation can enable the partial automation of classification decisions while only assisting the ML models for the most uncertain instances. This approach saves a significant amount of human resources compared to fully manual classification. In addition, the overall classification performance is expected to increase. One field of application is the filtering of offensive and unwanted comments in online forums [297], which is usually still performed manually due to the lack of highly accurate ML models. Active Moderation can drastically reduce the moderation overload and free up human resources. Human moderators can then focus on less repetitive tasks while still achieving higher classification performance than either machines or humans alone can accomplish. For example, forum moderators can moderate the most uncertain instances to the extent that is actually manageable.

Another use-case is to enhance the applicability of green ML solutions for real-world use-cases. For example, computational-aware Active Moderation can be applied to classify issue types in app stores without significantly increasing the carbon footprint. The lower prediction performance of lightweight classifiers is compensated for by manually correcting the most uncertain predictions with a manageable amount of effort. As a result, reliability is greatly improved.

A computationally efficient approach to Active Moderation is fundamental to bringing highly accurate classifiers closer to the actual users of the systems, thus facilitating deployment on edge devices. Many HiL tools, such as Forum 4.0 [134], enable human users to rapidly build personalized prototype ML models on their own workstations. Since our framework demonstrates practicality even with 4 GB of main memory, it can be easily integrated into resource-constrained workstations. Furthermore, lightweight ML models pave the way for implementing other HiL patterns that rely on low model latency. Generally, Active Moderation also facilitates the agile deployment of ML systems. Tuning ML models to achieve a desirable level of performance is challenging. Active Moderation enables the use of less refined models at earlier stages of their development during deployment. This significantly reduces the time needed to obtain an executable ML system.

Overall, the applicability of computational-aware Active Moderation depends on whether human effort is affordable during the operational use of a classifier and whether more reliable classification outcomes are actually needed.

5.5.3 Threats to Validity

The following section outlines the internal and external threats to validity.

Internal Threats. Our benchmark experiments rely heavily on the accuracy of the labeled data we use for training, testing, and validation. Since most of the datasets are labeled by humans, annotation errors, inconsistencies, or subjectivity may introduce biases that affect the reliability of the labels. To mitigate this limitation, we rely on established datasets from the literature.

We also simulate human labeling due to a lack of capacity to manually label thousands of comments. It is rather unlikely that a real human would randomly mislabel instances with a probability that is independent of the text being labeled. Simulated human participation may not fully reflect and represent the behavior of real humans. However, simulating human labeling for the sake of consistency and standardization is a common pattern for evaluating HiL approaches [94, 154, 221]

Given the limited number of classifiers and encodings used in our study (nine and two, respectively), it is possible that we missed a configuration that could have produced better results. Expanding the benchmark to include additional classifiers or fine-tuning the parameters could optimize their classification performance or reveal better results.

External Threats. Data labeling typically requires a considerable investment of time and human resources. The high classification performances showcased in our study might prove impractical if the associated manual effort is not affordable or if human resources are scarce. For instance, while computational-aware Active Moderation may attain a certain level of classification performance, this could surpass the capacity of manual efforts. Effectively navigating these constraints and establishing pragmatic expectations is crucial to deploying Active Moderation in practical settings.

We conducted the Active Moderation experiments on a limited number of datasets. However, the effectiveness and results of Active Moderation may not directly generalize to other datasets or domains due to variations in data characteristics, task complexity, or the presence of domain-specific biases. The generalizability of our findings raises concerns about their validity. We sought to include diverse domains and datasets in our analysis to mitigate these concerns.

The effectiveness of Active Moderation may vary depending on the characteristics of the dataset faced during deployment. Certain dataset characteristics, such as class imbalance, noise, or concept drift, can challenge Active Moderation and limit its performance. Our experiments used random splits to train, validate, and test our framework from the same data distribution. During deployment, the data will likely shift over time, resulting in a performance degeneration. Therefore, results may differ in real-world use-cases. Understanding and accounting for these dataset-specific limitations is critical.

Furthermore, we have only considered the scenario where a batch of already collected data needs to be classified at once. In the real world, however, user comments are not only historical, but may also come as a continuous stream of data. In this case, it is impractical to assign a specific portion of the data to a human, since the exact amount of comments is unknown.

5.5.4 Alternative Approaches from Related Work

The literature discusses several alternative approaches to reducing the number of misclassifications, mostly following the patterns of Active Moderation or Safeguards.

Lee et al. [216] propose a similar approach of detecting whether an instance is out-of-distribution from the training dataset and therefore likely to be misclassified. However, their approach requires an auxiliary dataset representing out-of-distribution samples during training, which is difficult to create. De et al. [83] introduce a semi-automated approach that directly optimizes a classifier for different levels of automation. However, their approach only applies to convex margin classifiers such as SVMs. Xiao et al. [393] propose a self-checking mechanism for NNs, where the features of the internal layers are used to check the reliability of the predictions. In contrast, our framework uses prediction uncertainties obtained via a softmax function, which is relatively easy to implement.

Another strategy to mitigate error-prone classifications is to allow classifiers to abstain when clear decisions cannot be made [74, 303]. This can be achieved by introducing an additional label into the classification task or by training a separate classifier. Abstained instances could then be referred to human moderators, similar to our approach. Abstaining from making a prediction typically entails a binary decision. Bilgic and Getoor [36] suggest an approach to misclassification detection based on the premise that misclassified instances are likely to have similarly misclassified neighbors or are most similar to other misclassified instances. Their method revolves around identifying clusters of misclassifications and comparing them to incoming new instances.

5.6 Conclusion

This chapter has investigated the suitability of an Active Moderation-based framework to differentiate between cases where human intervention should and should not be required. In particular, it has focused on providing highly accurate yet applicable text classification solutions for low-end infrastructure where strong state-of-the-art models would not be applicable due to resource constraints. The main findings are outlined below.

- Traditional ML models with confidence-based uncertainty estimates can effectively implement Active Moderation. Most classifiers are highly effective at detecting when a prediction is likely to be incorrect.
- Computational-aware Active Moderation can achieve high levels of accuracy even with lightweight classifiers. Specifically, the proposed framework increased the F1 score from approximately 78% to at least 95% on five out of six datasets by manually validating less than 29.6% of the data.
- An actively moderated classifier has the potential to be more accurate than a purely automated or purely manual classifier, even when only a noisy labeler is available.
- There is a high variability in the classification performance of Active Moderation depending on the dataset and classifier used. LR and MLP were the best-performing classifiers regarding computation time and F1 score for implementing computational-aware Active Moderation in our experiments.
- Active Moderation can be effectively applied to lightweight text classification models. Training and inference took only 1 second with only 4 GB of main memory, making it highly computationally efficient.
- Bayesian modeling, such as MCD, only slightly improved the quality of the estimated uncertainty. It did not have a major impact on the resulting F1 score of an actively moderated classifier using a small MLP.

Chapter 6

Human-resource-aware Active Moderation

Publication. This chapter is based in part on the 2022 paper "Efficient, Uncertainty-based Moderation of Neural Networks Text Classifiers" [11]. My contribution includes the conceptualization and implementation of the human-resource-aware Active Moderation framework, including the modeling and quantification of prediction uncertainties in various neural text classifiers. I also conducted the experiments, analyzed the results, and led the writing of the paper.

Contribution. We propose a framework for the human-resource-aware Active Moderation of text classifiers. In Chapter 5, we have shown that the Active Moderation pattern effectively improves the classification performance of already trained lightweight text classifiers during deployment. In this chapter, we address the challenge of maintaining cost-efficiency regarding the involvement of human effort during the moderation process. Further, we focus on deep learning models. We introduce a saturation-based moderation strategy for Active Moderation to maintain an optimal balance between classification performance and human effort. The goal is to reach a highly human-resource-aware moderation process while still achieving top accuracies. Across a spectrum of ML experiments, we observe a decrease in the efficiency of Active Moderation as the human workload increases. In addition, our findings reveal that there is a specific point in the moderation process where it is not worthwhile for a human to continue moderating due to the increasing sparsity of classification mistakes. Leveraging the proposed human-resource-aware Active Moderation framework, we attained top accuracies of up to 98–99%, while limiting human involvement to about one-third to one-fourth of the data. Furthermore, a noisy annotator who guesses with a probability of 15% when labeling only reduces the F1 score by about 3%. The proposed framework is applicable to any classifier that provides accurate uncertainty estimates.

6.1 Motivation

Achieving top classification performance in ML-based text classification is difficult, and compromises must be accepted. Even the most advanced state-of-the-art text classifiers often fail to achieve classification performance that can be considered applicable to real-world use-cases. For example, Hey et al. [149] achieve an F1 score of 73% to 93% using a state-of-the-art BERT model for requirements classification, which may be too low for use in production. ML-based classifiers cannot be used as a stand-alone solution if a much higher classification performance is desired. Without sufficient support from ML models, service providers may have to restrict their services to limit the amount of data to a level that can be handled manually [285].

We have previously identified Active Moderation as a promising HiL pattern to reduce the number of incorrect deployment decisions (misclassifications). Using Active Moderation, we successfully increased the overall applicability of lightweight text classifiers on low-end infrastructure (Chapter 5). The prediction uncertainty was shown to be sufficient to effectively identify misclassifications. A key finding of our investigation was that the more effort a highly experienced human was willing to put into correcting predictions, the higher the overall classification performance. However, HiL approaches that rely heavily on human annotators face another critical bottleneck: human effort. Since human resources are costly and do not scale well to larger workloads, moderation processes should be designed with human resource efficiency in mind. However, the efficient in-operation integration of human effort for building semi-automated decision systems remains largely unexplored. Furthermore, we have only studied lightweight models that do not allow for expressive Bayesian modeling. It remains unknown how Active Moderation performs on deep NNs.

By default, Active Moderation lacks the ability to be cost-efficient in terms of human effort. We use the term "**human-resource-aware**" to describe the cost-efficient use of human resources, i.e., optimizing the number of times a human is consulted to label a text instance. It remains unknown how to solve the trade-off between manual effort and achieved classification performance in a human-resource-aware manner. A naive approach to determine the number of instances to manually label according to a given annotation budget has several shortcomings. If too little human effort is spent, much potential is lost. If too much effort is spent, a disproportionate amount of effort is required for a comparatively small gain in classification performance. Finding an optimal threshold between classification performance and human effort would greatly improve the efficiency and applicability of Active Moderation.

6.2 Conceptual Framework

This section outlines human-resource-aware Active Moderation for text classification. We define the problem statement and present our saturation-based moderation strategy, which aims to maintain high human efficiency during the moderation process.

6.2.1 Problem Statement

In Chapter 5, we found that moderating the k most uncertain instances of a dataset significantly improves classification performance. The question remains: which k provides the most human-resource-aware moderation while maximizing the overall classification performance? The term human-resource-aware refers to the careful and economical use of human resources. Humans should only be involved in the classification process when their intervention is necessary, i.e., when a correction is to be made. If no correction is needed, but the human is still involved, there is no added value to the overall classification performance. However, using a fixed k as a separation criterion is not flexible and would not adapt to the individual characteristics of the variety of real-world data. A much more versatile approach is required.

To provide a human-resource-aware solution to the Active Moderation problem, we aim to separate all text instances into those that are better decided manually and those that an ML-based classifier can handle well on its own. Similar to the framework introduced in Section 5.2, the search for an optimal separation criterion can be formulated as follows: a partition of the data X to be classified into two discrete sets X_H and X_A is sought, with $X_H \cup X_A = X$ and $X_H \cap X_A = \emptyset$. The set $X_H \subseteq X$ represents all elements that should be moderated by humans. X_H should consist of the most uncertain and probably misclassified instances. On the other hand, $X_A \subseteq X$ describes the set of instances that should be classified automatically. A separation criterion is needed to determine in a human-resource-aware manner whether a new element $x \in X$ to be classified should belong to X_A or X_H when the classifier is used in production.

For human-resource-aware Active Moderation, a moderated classifier $f^{\omega}_{mod}(X)$ is created from the manual and automated predictions: $f^{\omega}_{mod}(X) := o_H(X_H) \cup f^{\omega}(X_A)$, where o_H is a human oracle and f^{ω} an ML-based classifier. To ensure the applicability of f^{ω}_{mod} over purely automated approaches f^{ω}, the following conditions must be met:

- Each input x to be classified is either an exclusive part of X_A or X_H. Thus, each input is finally classified manually or automatically.

- The classification performance of the automated classification $\hat{Y}_A \approx f^{\omega}(X_A)$ should be as high as possible.
- Keep the number of elements in X_H as small as possible to ensure the efficiency of human involvement.
- Human classifications $o_H(X_H)$ should be much more accurate than automatic classifications $f^{\omega}(X_H)$.

Given these conditions, a heuristic is needed to efficiently determine whether an instance x to be classified should rather belong to X_A or X_H.

6.2.2 Human-resource-aware Active Moderation as a Deployment Pattern

A human-resource-aware moderation of classification outcomes aims to save human resources while optimizing the overall classification performance during model deployment. We have previously shown that predictions with high uncertainty have a higher error rate than predictions with very low uncertainty. Thus, delegating highly uncertain classifications to a human oracle can significantly reduce the number of misclassifications. Instead of relying on a fixed value of k, we determine whether an instance should undergo manual moderation based on an uncertainty threshold $\vartheta_u \in \mathbb{R}^+$. We propose to create a human-resource-aware moderated classifier f^{ω}_{mod} from an artificial classifier f^{ω} as follows:

$$f^{\omega}_{mod}(x) := \begin{cases} f^{\omega}(x) & \text{if } u[y|x,\omega] \leq \vartheta_u \\ o_H(x) & \text{else} \end{cases} \tag{6.1}$$

where $o_H : X \rightarrow Y$ represents the human oracle, $u[y|x,\omega] \in \mathbb{R}^+$ an uncertainty measure of $f^{\omega}(x)$ and ω the learned parameters of f. If the uncertainty is below a threshold ϑ_u, the inferred label $y = f^{\omega}(x)$ is considered reliable and is kept. If the threshold is exceeded ($u[y|x,\omega] > \vartheta_u$), a human oracle o_H is consulted and its decision is used as the classification outcome.

6.2.3 Saturation-based Moderation Strategy

We propose a **saturation-based** moderation strategy to determine the uncertainty threshold ϑ_u. Since misclassifications are more frequent at high uncertainty levels, the classification performance of an actively moderated classifier becomes less efficient as the moderation load increases. At some point, significant improvements may no longer be achievable, and further effort will yield diminishing improvements in classification performance. A saturation-based

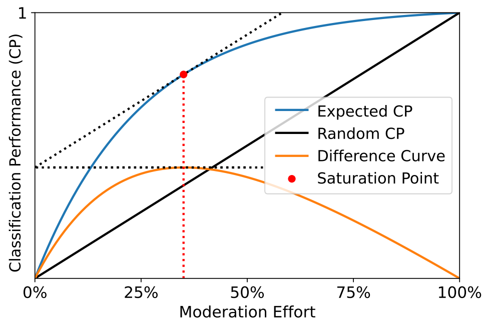


Figure 6.1: Saturation detection for manual moderation. Saturation is reached at the highest point of the difference between the expected and random classification performance curves.

strategy attempts to restrict the moderation to a point where the expected improvement diminishes, indicating that the cost of continued moderation outweighs the gain in classification performance.

Figure 6.1 depicts a hypothetical saturation curve of human-resource-aware Active Moderation. The blue curve illustrates the expected classification performance of our framework when a certain number of the most uncertain predictions are manually moderated. The classification performance of a moderated classifier relies on the manual annotations and the outcomes of the ML model classifying the instances that are not forwarded to a human. In an ideal setting, a classification performance of 100% is achieved when 100% of the instances are decided manually. In practice, however, achieving a classification performance of 100% manually would be unrealistic. ML models and humans are prone to making mistakes [45, 148]. Nevertheless, it is generally assumed that humans, especially domain experts, are capable of producing ground truth data labels. The expected classification performance curve follows the shape $f(x) = a(1 - \mathrm{e}^{-bx})$. The black line represents the classification performance of Active Moderation when instances are randomly sampled for moderation. Since a random sample of instances is expected to contain the same proportion of misclassifications as the entire dataset, the gain in classification performance increases linearly.

A natural **point of saturation** can be calculated as the peak of the difference curve between the expected and random classification performance [324]. In general, a saturation point describes a scenario in which the cost of increasing a parameter is no longer justified by the corresponding performance improvement. In the context of Active Moderation, the saturation point describes the situation where a continued uncertainty-based moderation becomes less effective than moderating randomly selected instances. At this point, correcting one additional misclassification is not worth the effort. We argue that moderation should stop at this natural limit in order to maintain human-resource-awareness.

6.2.4 Uncertainty Assessment

To estimate epistemic and aleatory uncertainties in f^{ω} we use techniques that have shown efficiency in similar uncertainty-driven tasks, such as computer vision [188] and Active Learning studies [57, 114]. We focus on three uncertainty modeling techniques and a baseline. Specifically, we apply the uncertainty modeling techniques MCD, BBB, and Ensemble (as introduced in Section 2.2.5) along with a baseline. The baseline uses deterministic softmax outcomes as an indicator of confidence [148].

Furthermore, we utilize score functions based on the uncertainty modeling techniques and a baseline to quantify the uncertainty of individual predictions. For our experiments, we consider commonly used score functions, namely *least confident*, *smallest margin*, and *mutual information* (Section 2.2.6).

6.3 Study Design

We continue our investigation of Active Moderation, with an emphasis on human efficiency. Our goal remains to develop a domain-independent framework that can be applied to various real-world text classification tasks. Thus, we persist in examining diverse domains in our analysis of human-resource-aware Active Moderation. We outline the research questions, study design, datasets used and details of our implementation of human-recourse-aware Active Moderation.

6.3.1 Research Questions

We aim to address the following research questions:

RQ1: How does uncertainty modeling improve the classification performance of unmoderated classifiers?

RQ1 investigates whether current deep learning-based text classifiers with uncertainty modeling (still a fully automated classification without moderation) can effectively address the problem by achieving a top classification performance close to 99%. In this case, an ML model could sufficiently solve the classification problem on its own and there would be no need for human intervention.

RQ2: How efficient is Active Moderation in terms of misclassification detection rates?

Human efficiency is critical to the design of a HiL deployment pattern. With RQ2 we investigate how efficient our uncertainty-based Active Moderation framework actually is. Efficiency is measured by the ratio of correctly classified to misclassified instances referred to a human moderator.

RQ3: How much would saturation-based moderation improve classification performance, and at what human cost?

In RQ3, our objective is to evaluate the human-resource-awareness of our framework. We empirically investigate how our proposed saturation-based moderation strategy resolves the trade-off between achieved classification performance and manual effort and whether top classification performance is still achievable.

RQ4: How do noisy oracles affect the classification performance of a human-resource-aware moderation?

Humans annotators are imperfect and prone to unintentional errors, clearly impacting the accuracy of manual annotation. In RQ4, we assess the influence of noisy annotators on our human-resource-aware Active Moderation framework.

6.3.2 Benchmark Criteria

To answer our research questions, we perform a series of ML experiments. First, to answer RQ1, we evaluate the initial classification performance of the three classifiers augmented with the different uncertainty modeling techniques and the baseline across all three datasets. We use the micro F1 score to measure the classification performance of the classifiers on the actual classification task without manual moderation. Additionally, we evaluate the ability of the classifier to detect misclassifications by computing the AUC-ROC score [81], based on the correct (positive class) and misclassified (negative class) outcomes. In this setting, the AUC-ROC score can be interpreted as the probability that a correct classification will receive a higher confidence score than a misclassification. For RQ2, the mean confidence scores of all correct and misclassified outcomes are measured. Second, to evaluate the human efficiency of Active Moderation, we assess the ratio of misclassifications assigned to humans for moderation relative to a given moderation load. Regarding RQ3, we evaluate the gains in classification performance of human-resource-aware Active Moderation and determine the most appropriate uncertainty estimation technique and model setting. We compute saturation points to estimate the achievable F1 scores while limiting human moderation effort. Finally, to address RQ4, we assess the impact of human noise; we simulate human failures by allowing the oracle to make guesses given a certain noise level. We employ noise levels of 0%, 5%, 10%, and 15%, where a class label is randomly selected from all possible classes given the noise level. We investigate the F1 score achieved across all datasets when a noisy human moderates instances according to the suggested saturation points. All experiments are performed on the hold-out test dataset.

6.3.3 Datasets

To experimentally evaluate our saturation-based moderation framework, we conducted several benchmarking experiments using three publicly available datasets. These are summarized in Table 6.1.

Dataset	Size	$\lvert C\rvert$	Class Distribution	#Words ($\mu \pm \sigma$)
Hate Speech	40,000	2	23,775:16,225	63 ± 104
IMDB	50,000	2	25,000:25,000	234 ± 173
20NewsGroups	18,846	20	999:997:996:994:991:990:990:988:987:985: 984:982:975:973:963:940:910:799:775:628	315 ± 658

Table 6.1: Statistics about the datasets used in the benchmark experiments.

The **Hate Speech** dataset, provided by a recent Kaggle competition[1], consists of Wikipedia comments that have been manually labeled for toxic behavior. We unify different forms of toxicity in the dataset into a binary classification task (*toxic* and *non-toxic*) and conduct our experiments on a random subset of 40,000 comments. The **IMDB** dataset [243] consists of 50,000 highly polarized English movie reviews associated with either a *positive* or *negative* sentiment. Lastly, the **20NewsGroups** dataset [212] consists of 18,846 English documents grouped into 20 different news topics. For the training, validation, and test sets, we randomly sampled from the Hate Speech, IMDB, and 20NewsGroups datasets using train-validate-test splits of 20,000:10,000:10,000, 25,000:12,500:12,500, and 9,846:4,500:4,500, respectively. We conducted all experiments five times with randomized data splits.

6.3.4 Implementation Details

In our evaluation of human-resource-aware Active Moderation, we utilize three common NN architectures from the literature: CNN, KimCNN, and DistilBERT. The **CNN** consists of a convolutional layer, a global max-pooling layer, and two fully connected dense layers, similar to recent studies on app reviews and tweet classification [350]. Dropout is applied before each weight layer at a rate of 0.4 with L2-regularization using a kernel penalty of 10^-5. As feature representations, we employ 100-dimensional trainable word vectors that

[1] `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge`

are randomly initialized. For **KimCNN**, we follow the NN architecture and configuration recommended by Kim [195]. The author proposes a multi-channel convolutional NN with different filter region sizes, followed by a 1-max pooling layer. As feature representations, we use static 300-dimensional Google word2vec embeddings, which are pre-trained on 100 billion news articles [253]. Finally, we employ the widely adopted state-of-the-art text classification model **DistilBERT** [323], a distilled version of BERT. DistilBERT incorporates 40% fewer parameters than BERT, making it significantly more efficient for training while maintaining approximately 97% of its classification performance. We fine-tune DistilBERT using the default configurations of the Hugging Face Trainer API[2].

For the uncertainty modeling techniques MCD and BBB, 50 stochastic forward passes are used. We consider 5 NNs as the size of the ensembles. It has been shown that larger ensembles do not significantly improve the uncertainty estimates [210]. For the MCD and baseline approaches, we perform inference on the same trained ML model since they share the same training procedure. For the implementation of BBB in the CNN and KimCNN, we replaced the network layers with Bayesian layers using the TensorFlow Probability[3] library. For DistilBERT, we only implement MCD by activating the model's internal dropout layer at inference time, as done by Miok et al. [256]. BBB cannot be applied directly to DistilBERT. This would require changing the network architecture and re-training DistilBERT from scratch.

We utilize the Kneedle algorithm [324] to detect the saturation point, as discussed in Section 6.2.3. Since real saturation curves often lack smoothness, we employ polynomial interpolation to fit a spline for saturation point detection.

6.4 Results

We outline the results of our experiments and answer the research questions. Our replication package is publicly available online[4].

6.4.1 Impact of Uncertainty Modeling on the Classification Performance

Table 6.2 presents the initial F1 scores of the classifiers when no manual moderation is performed. The table is structured according to the evaluation metrics

[2] `https://huggingface.co/transformers`
[3] `https://www.tensorflow.org/probability`
[4] `https://github.com/jsandersen/CMT`

	Hate Speech				IMDB				20NewsGroups				
Metrics	Baseline	MCD	BBB	Ensemble	Baseline	MCD	BBB	Ensemble	Baseline	MCD	BBB	Ensemble	
F1 score ↑	90.2|0.1	**90.6**|0.2	90.4|0.2	90.4|0.3	88.7|0.1	89.0|0.1	89.0|0.1	**89.6**|0.2	86.9|0.2	87.1|0.1	87.4|0.3	**90.1**|0.4	**CNN**
Mean Conf. Mis.	89.3|0.2	83.4|0.4	87.5|0.3	85.2|0.4	88.5|0.6	79.5|0.2	82.5|0.7	82.5|0.2	65.4|0.7	57.5|1.9	57.2|0.8	55.5|0.8	
Mean Conf. Cor.	98.3|0.1	96.9|0.1	98.1|0.1	97.6|0.1	97.9|0.1	95.2|0.1	96.3|0.1	96.3|0.1	94.4|0.3	85.1|0.4	91.6|0.2	90.8|0.2	
Range ↑	8.7	**13.5**	10.6	12.4	9.4	**15.7**	13.8	13.8	29.0	34.6	34.4	**35.3**	
AUC-ROC ↑	86.3|0.5	86.1|0.6	86.3|0.4	**86.8**|0.3	83.7|0.2	**84.0**|0.1	83.8|0.4	83.8|0.2	89.8|0.1	90.0|0.2	**90.2**|0.4	89.2|0.5	
F1 score	91.2|0.2	**91.4**|0.1	91.3|0.2	91.3|0.1	88.9|0.2	**89.7**|0.2	89.3|0.1	89.5|0.1	88.2|0.3	88.7|0.2	86.8|0.2	**89.5**|0.2	**KimCNN**
Mean Conf. Mis.	82.6|0.8	77.5|0.3	82.8|0.3	81.7|0.2	78.5|0.5	73.4|0.2	82.3|0.3	76.5|0.5	54.2|0.5	48.1|0.4	60.1|0.3	52.0|0.4	
Mean Conf. Cor.	97.0|0.3	95.4|0.1	97.1|0.1	96.8|0.1	97.5|0.1	92.2|0.1	96.5|0.1	93.6|0.2	90.8|0.3	84.5|0.4	92.0|0.1	89.5|0.2	
Range	14.4	**17.9**	14.3	15.1	15.9	**18.8**	14.2	17.1	36.6	36.4	31.9	**37.5**	
AUC-ROC	85.1|0.6	**87.6**|0.2	85.2|0.4	85.9|0.0	83.6|0.4	84.7|0.3	**85.0**|0.2	84.0|0.4	88.4|0.3	**89.1**|0.3	88.5|0.5	88.2|0.2	
F1 score ↑	94.0|0.2	**94.1**|0.1	-	94.0|0.1	93.7|0.1	93.7|0.1	-	**93.9**|0.2	90.5|0.4	90.4|0.4	-	**91.1**|0.3	**DistilBERT**
Mean Conf. Mis.	86.6|0.5	83.3|0.6	-	85.8|1.2	85.7|1.1	82.1|0.8	-	82.8|0.8	71.1|1.7	66.4|0.9	-	68.3|0.8	
Mean Conf. Cor.	98.5|0.1	98.1|0.1	-	98.5|0.1	98.2|0.3	97.5|0.2	-	97.7|0.2	95.1|0.2	93.5|0.2	-	94.5|0.1	
Range ↑	11.9	**14.8**	-	12.7	12.5	**15.4**	-	14.9	24.0	**27.1**	-	26.2	
AUC-ROC ↑	89.5|0.5	**91.6**|0.3	-	91.4|0.4	88.7|0.4	88.9|0.4	-	**89.0**|0.3	90.2|0.4	**90.4**|0.3	-	**90.4**|0.3	

Table 6.2: Effect of extending NN text classifiers with the uncertainty modeling techniques (**without manual moderation**). Each cell displays the mean | standard deviation of five independent classification runs. For each of the nine experiments (3 classifiers and 3 datasets), the scores of the best and worst performing uncertainty modeling techniques are highlighted in green and red, respectively.

and the uncertainty modeling techniques employed for each dataset and classifier. Each cell contains the mean and standard deviation derived from five independent classification runs. For each classifier and dataset, the results of the best and worst performing uncertainty modeling techniques are highlighted in green and red, respectively. *Mean Conf.* represents the mean confidence score of all misclassifications *(Mis.)* and correct classifications *(Cor.)*. The mean confidence range (*Range*) is calculated as the range between the mean confidence scores.

The results indicate that CNN and KimCNN achieve similar F1 scores across all experiments, ranging from 86.9% to 91.4%. DistilBERT demonstrates superior classification performance, especially in binary classification tasks (Hate Speech and IMDB), with F1 scores up to 94.1%. However, even DistilBERT has room for improvement, as it falls short of achieving top F1 scores of around 99%. The results also show that the explicit modeling of uncertainty has only a small effect compared to the F1 scores of the baselines (less than 1%). The only exception is an ensemble model using CNN and 20NewsGroups (multi-label classification), which exhibits an improvement of 3.2%, reaching about 90%. Among the uncertainty modeling techniques, MCD performs best on the Hate Speech dataset, while an ensemble model performs best on the 20NewsGroups and IMDB datasets. BBB never achieves the highest F1 score.

The confidence scores show that explicit modeling of uncertainty leads to fewer overconfident misclassifications compared to the baseline. Uncertainty modeling tends to increase the confidence range between correct and incorrect classifications. Among the classifiers, KimCNN produced the least overconfi-

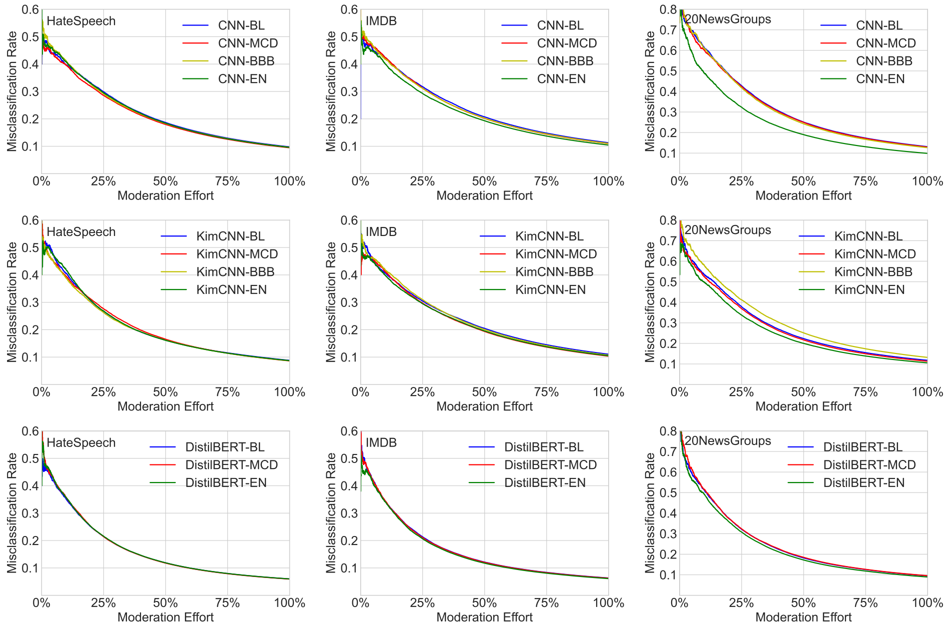


Figure 6.2: Human efficiency of Active Moderation. Efficiency is represented by the misclassification ratio when moderating a given number of the most uncertain instances.

dent false positives, followed by DistilBERT. Furthermore, our results reveal that MCD, BBB, and an ensemble generally outperform the baseline regarding misclassification detection (AUC-ROC). However, no specific technique consistently outperforms the others across all experiments.

6.4.2 Efficiency of Misclassification Detection

We examine the human efficiency of Active Moderation, which represents the ratio of misclassifications assigned to human moderation for a given moderation effort. Figure 6.2 illustrates the human efficiency using different uncertainty modeling approaches and the baseline.

Our results reveal that as moderation effort increases, human efficiency decreases. On the Hate Speech and IMDB datasets, the misclassification rate hovers around 50% when moderating less than 1% of the data. Thus, nearly every second moderation request is a misclassification. For the 20NewsGroups dataset, the misclassification rate is even higher, around 75%, when moderating less than 1% of the data. The efficiency curves illustrate a decline in human efficiency in resolving misclassifications. Misclassifications become sparser with increasing moderation effort. The misclassification rate converges to a classifier's classification performance at 100% manual effort.

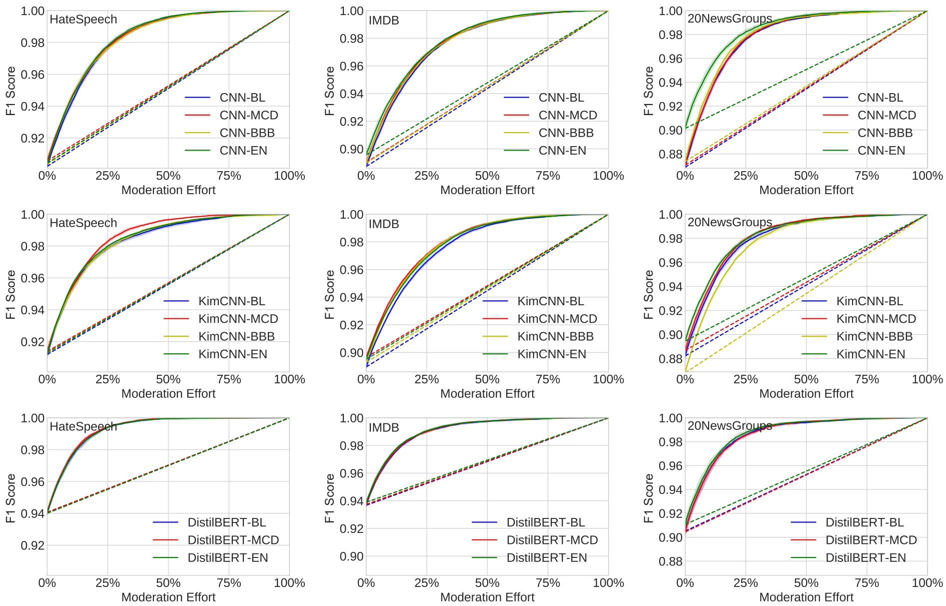


Figure 6.3: F1 score gains for the three considered datasets **with the proposed moderation** using different uncertainty modeling techniques and the baseline (BL). Dotted-lines illustrate the F1 scores of a random moderation strategy.

With a moderation effort of 10%, 20%, and 50%, the misclassification rate decreases to 41%, 32%, and 18%, respectively, on the Hate Speech dataset. There are no significant differences between the different uncertainty modeling approaches on the Hate Speech and IMDB datasets. On the 20NewsGroups, the ensemble approach is much less efficient to moderate. Of all the classifiers, DistilBERT is the least effective at detecting misclassifications.

6.4.3 Saturation-based Moderation Performance

Next, we examine the overall F1 scores of the human-resource-aware Active Moderation framework. This is the F1 score that is achieved when a specific proportion of the most uncertain instances are decided manually, while an automated classifier decides the remaining. Figure 6.3 illustrates these F1 scores for the Hate Speech, IMDB, and 20NewsGroups datasets. The y-axis represents the F1 score, while the x-axis indicates the corresponding manual moderation effort. In our first experiments, manual labeling is accomplished by selecting the ground truth label. Only the *least confident* score function[5] is reported for all experiments, as it achieves the highest F1 score in most cases and exhibits the most consistent classification performance.

[5] $unc_{LC}[y|x, D] := 1 - \max_c p(y = c|x, D)$

	Hate Speech				IMDB				20NewsGroups				
Saturation	Baseline	MCD	BBB	Ensemble	Baseline	MCD	BBB	Ensemble	Baseline	MCD	BBB	Ensemble	
Moderation Load	31.7	31.6	31.2	31.8	33.2	32.2	32.9	33.8	28.5	28.6	27.5	27.7	**CNN**
F1 score	98.08 (+7.9)	98.10 (+7.5)	97.94 (+7.6)	98.25 (+7.8)	97.80 (+9.1)	97.77 (+8.8)	97.77 (+8.8)	98.03 (+8.4)	98.10 (+11.2)	98.15 (+11.1)	98.13 (+10.8)	98.46 (+8.3)	**CNN**
Moderation Load	26.6	29.1	28.7	26.2	35.5	33.4	33.1	34.9	27.8	28.2	30.7	27.7	**K.CNN**
F1 score	97.57 (+6.4)	98.39 (+7.0)	97.77 (+6.5)	97.70 (+6.4)	98.01 (+9.1)	98.13 (+8.5)	98.12 (+8.8)	98.20 (+8.7)	98.00 (+9.8)	98.32 (+9.6)	98.04 (+11.2)	98.24 (+8.8)	**K.CNN**
Moderation Load	24.5	23.9	-	24.5	25.3	25.1	-	24.6	25.5	25.9	-	25.5	**D.BERT**
F1 score	99.36 (+5.3)	99.37 (+5.3)	-	99.37 (+5.4)	99.03 (+5.4)	99.01 (+5.3)	-	99.04 (+5.1)	98.60 (+8.1)	98.62 (+8.2)	-	98.82 (+7.7)	**D.BERT**

Table 6.3: Achieved F1 scores and moderation load required (in %) using our human-resource-aware Active Moderation framework.

Figure 6.3 depicts the gains in F1 score, highlighting a significant increase over a random moderation strategy, as shown by the dotted lines. Additionally, as demonstrated in Section 6.4.2, it can be seen that the moderation efficiency decreases with an increasing moderation load. Misclassifications become more frequent when a classifier reports large uncertainty scores.

A similar moderation behavior is observed across all examined classifiers. All F1 score curves for all classifiers adhere to the shape of a saturation curve, as we assumed (Section 6.2.3). The highest variations are observed on the 20News-Groups dataset. Moreover, the difference between all approaches diminishes with increasing moderation effort. By manually moderating more instances, all classifiers achieve more similar F1 scores. An ensemble of homogeneous NNs and MCD achieves the highest overall F1 score with the least moderation effort (with BBB slightly less effective). On average, the baseline requires slightly, but not significantly, more manual effort to achieve the same F1 score.

Since gains in classification performance decrease with an increasing moderation effort, we calculate saturation points to determine when to stop moderating before it becomes inefficient. Table 6.3 lists these saturation points for the LC score function. The absolute improvement in F1 score is provided in parentheses. The table demonstrates that our moderation framework can achieve F1 scores of 98% to 99% on all classification tasks while maintaining an efficient human moderation. These F1 scores can be achieved with all considered classifiers and uncertainty estimation techniques. Saturation points are reached after moderating less than 33.1% of the dataset for CNN and KimCNN and less than 25.5% for DistilBERT. All classifiers offer a similar trade-off between the F1 score achieved and the moderation effort. However, the baseline is sub-optimal as it either saturates with a slightly higher moderation effort or provides a lower F1 score than MCD, BBB and an ensemble. On the IMDB dataset, MCD reaches saturation with the least moderation effort while maintaining a high level of classification performance. Conversely, an ensemble performs best on

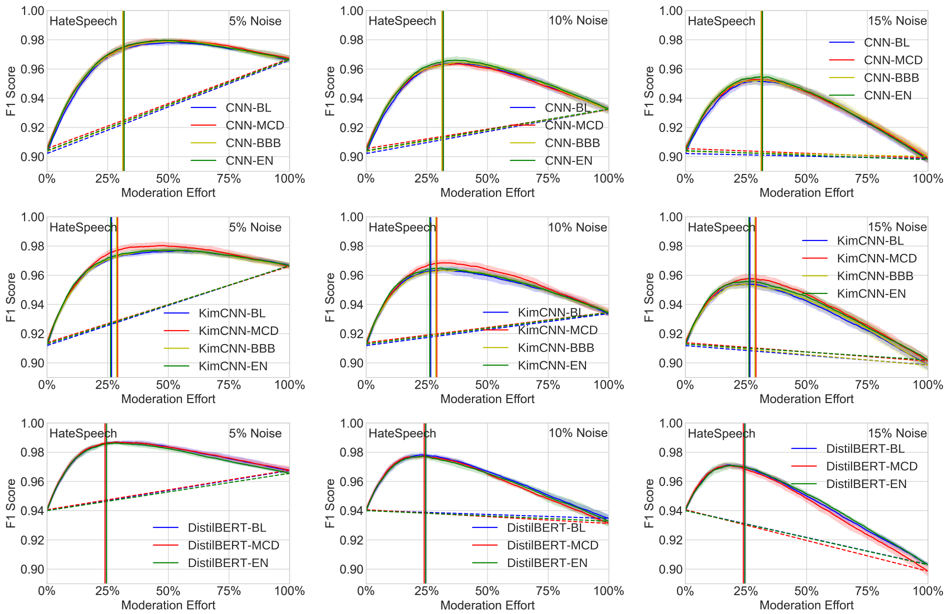


Figure 6.4: Active Moderation under human noise (5%, 10%, and 15%) on the **HateSpeech** dataset using various uncertainty modeling techniques. Vertical lines illustrate the saturation points.

the 20NewsGroups dataset. DistilBERT requires the least manual effort while achieving the highest level of F1 score, that is 98.6% to 99.37%. The results also reveal that models with low initial F1 scores achieve higher absolute F1 score improvements. Overall, a moderator using our framework has to label up to 73.3% (Hate Speech), 71.0% (IMDB), and 70.9% (20NewsGroups) *less data* instances compared to a random moderation strategy to attain the same F1 score.

6.4.4 Effect of Human Noise

Lastly, we investigate how a noisy human would affect the classification performance of the proposed saturation-based moderation framework. Figure 6.4 shows the F1 scores of the saturation-based moderation under different noise levels for the Hate Speech dataset. Figures A.1 and A.2 illustrate the results for the IMDB and App Store datasets. The vertical lines illustrate the saturation points. Since humans are now committing errors (random guesses in 5%, 10%, or 15% of cases), the classification performance does not peak at 100% when all data is manually labeled. In fact, after a certain amount of manual effort, the classification performance starts to degrade, as a point is reached where humans are making more labeling mistakes than the classifier. Thus, stopping the moderation is much more crucial. Clearly, the higher the noise level of the

	Hate Speech				IMDB				20NewsGroups				
Saturation	Baseline	MCD	BBB	Ensemble	Baseline	MCD	BBB	Ensemble	Baseline	MCD	BBB	Ensemble	
Moderation Load	31.7	31.6	31.2	31.8	33.2	32.2	32.9	33.8	28.5	28.6	27.5	27.7	**CNN**
F1 score 5% Noise	97.46 (+7.3)	97.35 (+6.8)	97.31 (+6.9)	97.50 (+7.1)	96.70 (+8.0)	96.70 (+7.7)	96.68 (+7.7)	96.92 (+7.3)	96.72 (+9.8)	96.82 (+9.7)	97.00 (+9.6)	97.17 (+7.1)	**CNN**
F1 score 10% Noise	96.31 (+6.1)	96.26 (+5.7)	96.22 (+5.8)	96.45 (+ 6.1)	95.54 (+6.8)	95.65 (+6.7)	95.56 (+6.6)	95.73 (+6.1)	95.53 (+8.6)	95.36 (+8.3)	95.36 (+8.0)	95.84 (+5.7)	**CNN**
F1 score 15% Noise	95.32 (+5.1)	95.33 (+4.7)	95.25 (+4.8)	95.41 (+5.0)	94.52 (+5.8)	94.64 (+5.6)	94.52 (+5.5)	94.70 (+5.1)	94.07 (+7.2)	94.12 (+7.0)	94.18 (+6.8)	94.42 (+4.3)	**CNN**
Moderation Load	26.6	29.1	28.7	26.2	35.5	33.4	33.1	34.9	27.8	28.2	30.7	27.7	**Kwon CNN**
F1 score 5% Noise	97.16 (+6.0)	97.69 (+6.3)	97.27 (+6.0)	97.18 (+5.9)	96.88 (+8.0)	97.08 (+7.4)	97.03 (+7.7)	97.00 (+7.5)	96.72 (+8.5)	97.02 (+8.3)	96.56 (+9.8)	96.83 (+7.3)	**Kwon CNN**
F1 score 10% Noise	96.32 (+5.1)	96.80 (+5.4)	96.28 (+5.0)	96.31 (+5.0)	95.73 (+6.8)	95.80 (+6.1)	95.93 (+6.6)	95.81 (+6.3)	95.51 (+7.3)	95.56 (+6.9)	95.02 (+8.2)	95.64 (+6.1)	**Kwon CNN**
F1 score 15% Noise	95.31 (+4.1)	95.89 (+4.5)	95.21 (+3.9)	95.54 (+4.2)	94.57 (+5.7)	94.87 (+5.2)	94.88 (+5.6)	94.77 (+5.3)	94.04 (+5.8)	94.28 (+5.6)	93.72 (+6.9)	94.02 (+4.5)	**Kwon CNN**
Moderation Load	24.5	23.9	-	24.5	25.3	25.1	-	24.6	25.5	25.9	-	25.5	**DistiBERT**
F1 score 5% Noise	98.55 (+4.6)	98.53 (+4.4)	-	98.51 (+4.5)	98.20 (+4.5)	98.17 (+4.5)	-	98.19 (+4.3)	97.34 (+6.8)	97.37 (+7.0)	-	97.71 (+6.6)	**DistiBERT**
F1 score 10% Noise	97.71 (+3.7)	97.84 (+3.7)	-	97.71 (+3.7)	97.37 (+3.7)	97.35 (+3.6)	-	97.40 (+3.5)	96.28 (+5.8)	96.16 (+5.8)	-	96.49 (+5.4)	**DistiBERT**
F1 score 15% Noise	96.91 (+2.9)	96.96 (+2.9)	-	96.88 (+2.9)	96.41 (+2.7)	96.46 (+2.8)	-	96.53 (+2.6)	94.92 (+4.4)	94.78 (+4.4)	-	95.35 (+4.3)	**DistiBERT**

Table 6.4: F1 scores and moderation load (in %) achieved using our framework under different levels of human noise.

human moderator, the less accurate the actively moderated classifier becomes. Furthermore, it can be observed that the moderation of DistilBERT is more challenging due to its relatively high initial classification performance. In such cases, the human moderator must be highly accurate to outperform the model.

Table 6.4 presents the F1 scores of the classifiers when the moderation is stopped at the saturation points under various noise levels. The saturation points correspond to those in Table 6.3. The results suggest that even in the presence of human noise, saturation-based moderation leads to significant improvements in classification performance in all experiments. A 5% noisy human would achieve an F1 score of at least 96–98%, a 10% noisy human would achieve 95-97%, and a 15% noisy human would achieve at least 94–96%. This represents an average 3% decrease in F1 score when considering a 15% noisy human compared to a perfectly accurate labeler. Furthermore, the results demonstrate that superior F1 scores can be achieved through our Active Moderation framework, even when a human is highly noisy. For example, a 15% noisy human would attain an F1 score of approximately 93.5%, while the machine is 91% accurate. When combined, an F1 score of up to 96.53% is obtained (IMDB).

6.5 Discussion

This chapter proposes a novel uncertainty-based framework for implementing the Active Moderation pattern in a human-resource-aware manner. We discuss the implications of our findings, areas of application, and limitations of our framework.

6.5.1 Implications

Our results indicate that explicit uncertainty modeling alone can hardly improve the classification performance of ML-based text classifiers. However, our human-resource-aware Active Moderation framework can substantially improve the classification performance and thus the applicability of both weak (CNN / KimCNN) and strong (DistilBERT) text classifiers. Our HiL framework requires only a fraction of the human effort (a quarter to a third) compared to fully manual classification, while achieving top classification performance levels of approximately 98–99%. Even when a human is noisy (e.g. guessing 15% of the time), our framework still achieves a classification performance of 95–97%. This translates into substantial savings in human resources. Furthermore, our framework is highly efficient at correcting misclassifications. In particular, in a binary classification task, 50% of the first 1% of human assignments are misclassifications. Human efficiency then begins to decline. Stopping moderation when it becomes inefficient is the main advantage of our framework. Furthermore, our results show that the framework works well with standard deterministic NNs (baseline). Although the classification performance improvement and effort minimization are not as good as with explicit uncertainty modeling, the baseline achieves similar top F1 scores, especially with DistilBERT. Thus, the cost of implementing and adapting the framework to existing classifiers is rather low. The additional cost of explicit uncertainty modeling should be weighed against the marginal improvements that can be achieved.

Clearly, the usefulness of our framework depends on the application scenario at hand. In particular, it is crucial to first investigate:

- Whether or not a top classification performance, of e.g. 99% is expected by users or not.
- Whether and how human moderation is applicable, and whether the moderation can be trusted.
- Whether the goal is to maximize the classification performance while minimizing human effort.

We believe that HiL classification approaches are particularly important in domains with a very large number of classifications and where classification mistakes are costly, for example, when user comments need to be moderated in a public debate space, such as comments in news outlets [40, 234] or in Wikipedia as in the Hate Speech dataset. Purely automated classification and analysis of inherently ambiguous text, e.g. reflecting human opinions or outlining novel ideas, will quickly reach its limits. Even humans may not be able to agree on a single label for complex text [314]. As our experiments have shown, most text instances can be accurately labeled by an ML model and do not require human effort. However, complex or ambiguous text may not be adequately handled by black-and-white thinking, and machines may not be able to make reliable classifications. By bringing a human into the loop, human creativity and reasoning can help solve such difficult tasks efficiently.

In moderation, additional data is continuously collected and can be used to occasionally re-train the classifier (Continuous Learning). Frequent re-training prevents the classification performance of the underlying classifier from decaying over time [260] and may (but does not necessarily) improve its classification performance [20]. Our results suggest that moderation would benefit from a higher initial classification performance as the amount of human involvement seems to be lowest here.

In addition, the saturation-based moderation strategy of human-resource-aware Active Moderation is applicable to online classification settings that deal with a continuous stream of data. For each received instance, it can be individually judged whether it should be manually labeled or not. In comparison, a fixed size moderation load requires all data to be available prior to moderation.

6.5.2 Field of Application

The proposed human-resource-aware Active Moderation framework addresses a significant obstacle of HiL systems by maximizing the utilization of human labelers. Without considering human efficiency, the applicability of Active Moderation would raise critical concerns. We offer a comprehensive solution to the trade-off between manual effort and classification performance that has broad applications across various domains. Its field of application overlaps with that of computational-aware Active Moderation (Chapter 5).

A central use-case of human-resource-aware Active Moderation is in settings where a high level of classification performance is desired, but cannot be achieved by ML models alone. A typical example is the moderation of online forums. Here, human moderators are typically confronted with a large amount of data that needs to comply with certain guidelines. Human-resource-aware Active Moderation allows for a highly efficient use of human resources by only asking for help when it is most effective in terms of classification performance. Another use-case for our framework is data-driven software engineering. App stores and issue trackers serve as valuable sources of information from a software engineering perspective. Users typically express feature requests, bug reports, or praise and dispraise, which provide valuable insights for software developers. Since automated classification of these types of insights is challenging, our framework would enable a human cost-efficient HiL solution to raise the classification performance and usability of these sources. Active Moderation mitigates the risk of misclassifications and the cost of corrections.

6.5.3 Threats to Validity

We discuss the internal and external threats to validity.

Internal Threats. Our framework is based on the assumption that the expected classification performance curve of a classifier can be used to estimate how it would perform in practice. This assumption depends on whether the dataset used for testing and the derived saturation point represent the real data distribution. In cases where this assumption is not reasonable, and the real data distribution is significantly different, the results could be adversely affected. For example, the operational data distribution and the saturation curve would need to be monitored and possibly adjusted. Further research is needed to investigate the impact of distribution changes on our framework.

In some parts of our experiments, we assume that human moderators do not commit errors. While a flawless moderator is generally assumed in the review of interactive ML approaches such as Active Learning [57, 114, 158], this assumption may not hold in all real-world scenarios [335]. However, domain expert annotations are usually considered to be more thoughtful and trustworthy than machine annotations, especially for difficult tasks such as classifying ambiguous text. For instance, human annotations are generally considered as the ground truth for many classification tasks and are used to initially label training data[221]. This assumption may have limitations in practice, as even humans can make mistakes [45]. However, it is unrealistic to assume that labeling mistakes are randomly distributed across the dataset and that they are independent of the complexity of the actual labeling task. Real humans may behave differently.

Furthermore, interactive ML approaches such as our framework face a scalability limitation. Even a small fraction of human involvement can lead to enormous manual effort when the data to be classified is very large. Our framework can limit human involvement to labeling 23.9–25.4% of the data to achieve a top F1 score. It is up to the human moderators to decide whether this effort is applicable and desirable. While 23.9% of 1,000 might be manageable for a human moderator, a third of 100,000 will still cause scalability problems.

External Threats. Our framework also faces the limitation that uncertainty estimation approaches are unable to identify highly certain classifications that are actually wrong (unknown unknowns) [26]. Thus, it is unrealistic to avoid all misclassifications without manually checking all the data. However, we have shown that our framework can efficiently identify most misclassifications, resulting in high F1 scores of around 98–99%.

Finally, as with any empirical evaluation, our results depend on the datasets, metrics, and settings employed. While we refrain from claiming the generalizability of the specific quantitative results, the diversity of datasets and classification models used gives us enough confidence in the overall trends observed for text classification. To improve the generalizability of our results, more classification tasks, datasets, and model architectures should be evaluated.

6.5.4 Alternative Approaches from Related Work

Some approaches to coordinating human involvement within Active Moderation or Safeguards have been discussed previously. To the best of our knowledge, we are the first to propose a human-resource-aware Active Moderation framework.

Rattigan et al. [305] initially investigated the objective of maximizing the accuracy of classifiers while limiting human efforts. However, they only focused on relational data, while our study concentrates on text. To limit human effort, Pavlopoulos et al. [285] propose searching for upper and lower confidence thresholds that maximize a classifier's classification performance when classification outcomes between them are manually moderated. However, this approach requires human moderators to determine how much data they are willing to moderate. An evaluation of the efficiency is not performed, resulting in a non-optimal moderation load in most cases. Geifman and El-Yaniv [119] propose a classification approach with a reject option that allows practitioners to set a desired level of risk. Their approach includes training a secondary rejector classifier. Similar to our framework, they aim to ensure a certain classification performance. In contrast, they do not focus on the efficiency of human intervention. Lin et al. [228] suggest a similar rejection approach based on a set-classifier that controls the class-specific prediction risks. The idea is to reject a prediction if the set classifier returns more than one label.

Others, such as He et al. [143], aim to improve the quality of uncertainty estimates to improve the detection of misclassifications and thus reduce human effort. However, they do not address the trade-off between manual effort and classification performance. Further, their approach only applies to deep NNs. Our framework can be applied to any classifier providing uncertainty estimates.

6.6 Conclusion

This chapter has contributed to the HiL paradigm by investigating the Active Moderation pattern in the context of deep learning. We have presented an Active Moderation framework for text classification to minimize unreliable and error-prone classification outcomes. Based on explicit uncertainty modeling, our framework attempts to prevent uncertain classifications by consulting human moderators in a human-resource-aware manner. At its core, the framework uses a saturation-based moderation strategy that limits the moderation load and ensures a highly cost-effective use of human resources. We briefly outline the main findings:

- Bayesian modeling applied to deep learning models alone does not significantly improve classification performance. Thus, unmoderated deep NNs may still fail to deliver the desired classification performance.
- Active Moderation is most efficient with small moderation effort. Its efficiency decreases as human effort increases. At a certain point, continuous human moderation does not lead to significant performance improvements.
- A saturation-based moderation significantly improves the classification performance of ML-based text classification systems. In all our experiments, we increased the F1 score from an initial value of about 89–94% to approximately 98–99% by manually moderating between one-third and one-fourth of the data.
- Saturation-based moderation can efficiently limit manual effort. It saves up to 70.9% to 73.3% of the effort compared to a random moderation to achieve the same F1 score.
- Clearly, the effectiveness of Active Moderation declines when humans commit errors during labeling. Despite the presence of noise, our saturation-based moderation framework still maintains high improvements in classification performance. Using a state-of-the-art model, Active Moderation with a 15% noisy human performs on average only 3% worse than a human without noise.
- The baseline of using softmax probabilities without explicit uncertainty modeling performed surprisingly well within Active Moderation. No significant differences were observed compared to state-of-the-art uncertainty modeling techniques. Thus, explicit uncertainty modeling may not be necessary for implementing human-resource-aware Active Moderation.

Chapter 7

Low-latency Active Learning via Proxy-based Data Sampling

Publication. This chapter is based on the 2023 paper "Towards Low-budget Real-time Active Learning for Text Classification via Proxy-based Data Selection" [15], for which I was the lead researcher, implementer, experimenter, and writer.

Contribution. We propose a low-budget Proxy-based Active Learning framework for text classification. Our framework aims to improve the applicability of the traditional Active Learning approach while exploiting the processing capabilities of state-of-the-art models such as BERT. Current Active Learning studies yield promising results but suffer from high model latency, making them less suitable for real-world use-cases. Long waiting times lead to low user experience and productivity, which negatively affects their applicability. We adopt a Proxy-based data selection strategy to speed up the traditional Active Learning process, which consists of multiple consecutive re-training and inference steps. This strategy integrates an additional lightweight model during the learning process. We conduct several experiments to demonstrate its effectiveness. Our proposed framework outperforms previous research in terms of both model latency and classification performance. We show that Proxy-based Active Learning improves the quality of the collected training data and increases the classification performance by up to 19.34% compared to using only a lightweight ML model for Active Learning. In addition, Proxy-based Active Learning outperforms Passive Learning by up to 15.30%.

7.1 Motivation

Active Learning has proven to be a highly effective training pattern, significantly reducing human effort [125, 154, 221]. However, Active Learning has a special temporal requirement to maintain its applicability in real-world settings [333]. To maintain a certain level of user experience, fast interaction cycles are much more critical for HiL approaches than their actual classification performance [8, 103]. Maintaining fast training processes is a core principle of HiL training patterns. **Low model latency** is particularly important for Active Learning, since each training iteration requires the model to infer the labeling of a typically large corpus of unlabeled data, as well as to frequently re-train a model from scratch. Current studies on Active Learning often overlook the time required to perform a learning iteration, which casts doubt on their practical applicability.

In the field of text classification, encoder-based LLMs such as BERT have significantly advanced the state-of-the-art classification performance in various classification tasks. They have become the preferred choice for achieving the highest classification performance. Although BERT has been shown to work well in combination with Active Learning in a laboratory environment [94], it consists of hundreds of millions of parameters. Complex models require very long training and inference times, negatively impacting the overall user experience [93]. For example, Dor et al. [94] report a latency for BERT using a pool of 7000 unlabeled instances of 1.4 minutes using a Nvidia® Tesla K80 GPU.

To maintain fast interaction cycles, less complex models are promising to reduce the latency of Active Learning. However, a well-known trade-off exists between model complexity and classification performance [292]. This is because more complex models are able to represent more sophisticated classification functions. Finding the right balance between classification performance and model latency is not simple. Model complexity is not necessarily linearly related to classification performance, and the runtime depends primarily on the production environment, not just the model. To maintain low model latency, a random selection strategy – also known as *Passive Learning* – is typically left, which would not cause any model latency during labeling. In addition, previous Active Learning research has mostly considered large labeling budgets of several thousand instances, which may not be applicable in practice (e.g., up to 7,200 [293] or even 25,000 [159] instances). Therefore, applicable learning frameworks for text classifiers should also be human-resource-aware.

7.2 Conceptual Framework

This section describes a Proxy-based Active Learning framework for text classification. We outline the problem statement and illustrate our Proxy-based Active Learning pipeline.

7.2.1 Problem Statement

Training highly accurate models given some data is a core objective of text classification. However, training data is usually not available and must first be labeled, which is tedious, costly, and labor-intensive. It is desirable to use human effort as efficiently as possible while still obtaining highly accurate models. Active Learning as a HiL pattern (Section 4.5.3) aims at minimizing manual labeling costs while maximizing classification performance.

Active Learning is an iterative, model-centric approach to retrieving training instances from human annotators by actively soliciting labels. It involves a large pool of unlabeled data instances $X_{pool} = \{x_i\}_{i=1}^{N}$ and a small initial training set of already labeled instances $D = \{(x_i, y_i)\}_{i=1}^{M}$. Typically, unlabeled data is available in much larger quantities than labeled data. The pool of unlabeled data X_{pool} is assumed to contain the candidates that can potentially be used for training. The goal of Active Learning is to determine which and how many elements from X_{pool} should be labeled in order to effectively train a classifier $f \in \mathcal{F}$. In Active Learning, the classifier itself selects the training instances according to the expected contribution of an instance to its own learning behavior. To do this, an acquisition function $\alpha : (X \times \mathcal{F}) \rightarrow \mathbb{R}$ is used to rank the instances. Their labeling is then queried from a human annotator, also referred to as a human oracle $o_H : X \rightarrow Y$. The human oracle is assumed to always provide the correct label $\hat{y}$ for a given instance x so that it can be used as training data in the next training iteration. A few labeled instances D are required to initially train the model, which affects the effectiveness of α in early iterations. The simplest approach to initialize D is to use some randomly sampled instances for each class.

Low latency is a critical consideration for HiL systems. In Active Learning, humans are asked to label a sequence of instances one by one. At each step, the loop pauses and waits for human feedback. For Active Learning to be practical, the user experience of the overall system is crucial, as it affects the willingness of humans to stay in the loop. Tolia et al. [361] suggest that the time between interactions should be no more than 1 second to maintain a high level of user acceptance and productivity. However, in Active Learning, inference is performed on each instance of the data pool D_{pool} to rank its usefulness according to the acquisition function α. Furthermore, each Active Learning iteration requires the model to be re-trained from scratch using the entire training corpus D. Since state-of-the-art classifiers are too slow for fast Active Learning cycles, practitioners must choose less complex models to meet time constraints or rely on Passive Learning.

We explore an approach to enhance the user experience of Active Learning for text classification by substantially reducing model latency between learning iterations, while taking advantage of the processing capabilities of state-of-the-art classifiers.

7.2.2 Data Selection via a Low-budget Proxy

The traditional Active Learning process employs the same classification algorithm for both data selection and deployment (self-selection). However, in many situations, self-selection is neither applicable nor desirable [362]. This is particularly evident in real-world domains where rapid interactive cycles are critical to seamlessly integrate humans into the ML loop [103].

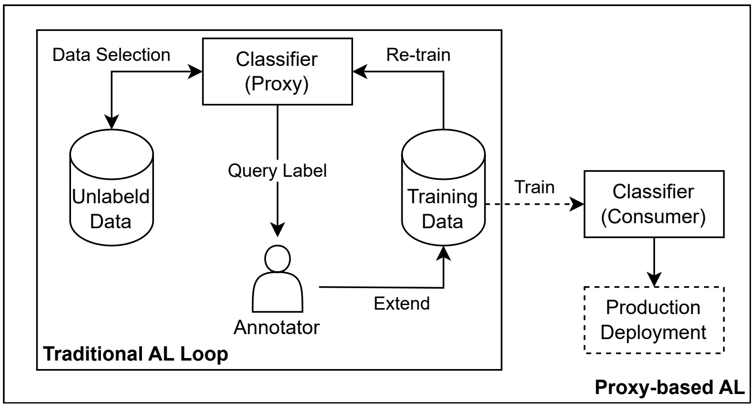


Figure 7.1: Proxy-based Active Learning is an extension of the traditional Active Learning process. In this approach, one classifier (Proxy) is used for data selection, while a second, typically more complex classifier (Consumer) is used for deployment.

In this section, we adapt Proxy-based Active Learning [71, 362] to enhance the applicability of text classifiers by increasing the computational efficiency of the learning process. Proxy-based Active Learning is a data-centric approach where the primary output is not a trained classifier, but a presumably high-quality training dataset that can be utilized to train a state-of-the-art classifier. Proxy-based Active Learning involves two distinct classifiers in the learning process. One classifier is dedicated to data selection (called the "*Proxy*"). The second classifier is trained on the acquired dataset (*Proxy sampling*) and used for deployment (referred to as the "*Consumer*"). To speed up the data selection process, the Proxy needs to be much faster, typically less complex, and generally provides a lower classification performance. In comparison, the Consumer is the model used in production. The Consumer should ideally be a state-of-the-art model that is likely to deliver the highest classification performance possible.

7.3 Proxy-based Active Learning for Text Classification

This section describes our implementation of Proxy-based Active Learning for text classification. To maximize the benefits of Proxy-based Active Learning, we leverage recent text classification techniques from the literature. We present candidate classifiers for both the Proxy and the Consumer and outline different data selection strategies for the Proxy.

7.3.1 Proxy Candidates

A Proxy candidate must demonstrate computational efficiency during both the training and inference processes, while delivering a reusable set of training data. A lightweight, low-latency classifier is essential for this purpose. In our investigation, we assess the suitability of **FastText** [176] and **Logistic Regression** (LR) as a Proxy. FastText was selected because speed and low complexity were pivotal to its development. LR was considered due to the promising results observed in our previous study on lightweight text classifiers (Chapter 5).

FastText. FastText is a simplistic NN-based linear text classifier that represents sentences by averaging trainable word vectors. Its model architecture consists of only two layers: a hidden layer, also known as the embedding layer, and a fully connected output layer. FastText operates as an end-to-end learning approach, seamlessly integrating feature engineering and classification. Figure 7.2 depicts the architecture of FastText.

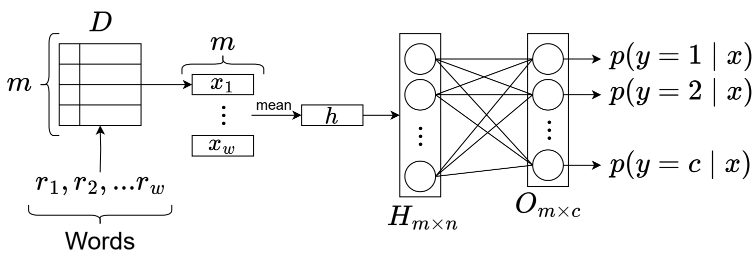


Figure 7.2: The architecture of the FastText model.

The input of FastText is a text document d represented by a vector of words $d = (r_1, r_2, ..., r_w)$ of size w. First, each word of d is replaced by a 1-hot-encoding scheme using a dictionary $\mathcal{D}$ of size m. Each word r is embedded by a feature vector $x = (x_1, x_2, ..., x_m)$ where x_j is set to 1 where j is the index of the word r in D. Other components of x are set to 0. Second, mean pooling is performed on the word embeddings to compute a m-dimensional distributed vector representation of d. Mean pooling averages all word embeddings x_i to a document embedding h, that is:

$$h = \frac{1}{w} \sum_{i=1}^{w} x_i \in \mathbb{R}^n \tag{7.1}$$

Third, the document embedding is fed into a fully connected hidden dense layer H to compute the output layer O. Finally, the averaged word representations are fed into a multivariate LR model. Classification is performed by computing the class probabilities of the output layer activation with the softmax function ϕ. Local optimal values for the weights W and O are estimated by minimizing the negative log-likelihood using backpropagation.

Experiments demonstrate that FastText performs comparably to simple deep learning-based approaches, while being an order of magnitude faster in both training and inference [176]. Due to its simplicity and effectiveness, FastText has emerged as a common baseline for text classification tasks. Another advantage is that FastText can yield effective results even with a limited amount of training data.

Logistic Regression (LR). In our previous investigation, we identified LR as a well-performing and lightweight text classifier (Chapter 5). We apply LR in combination with state-of-the-art embeddings as feature representations. We use pre-trained encoder-based LLMs without fine-tuning for feature extraction to save computational time. The embeddings are pre-computed for all data instances, cached, and queried during the Active Learning process. We consider the following state-of-the-art text embeddings from pre-trained encoder-based LLMs for feature extraction:

- **BERT**. We utilize BERT to extract contextual representations from text documents. As a feature representation, we employ the hidden state vector corresponding to the [CLS] token. In our implementation, we utilize the BERT base model, which converts textual input into a 768-dimensional feature vector.

- **SBERT**. We also consider SBERT for deriving text embeddings. The authors claim to provide out-of-the-box semantically meaningful embeddings that outperform pooled BERT embeddings on various classification tasks [308]. SBERT text embeddings have a dimensionality of 512.

- **ELMo**. Further, we employ *Embeddings from Language Models* (ELMo) [288], which is another approach to computing contextualized word representations. ELMo embeddings are derived from the internal states of a bidirectional language model. As a text embedding, ELMo has a dimensionality of 1024.

7.3.2 BERT Consumer

We deploy BERT as the Consumer in our investigation of Proxy-based Active Learning, as BERT has demonstrated state-of-the-art accuracies in numerous text classification tasks [88]. We add a softmax function on top of BERT using the *[CLS]* token as input to perform classification.

7.3.3 Selection Strategies

Several selection strategies for Active Learning have been investigated to sample the most informative text instances for manual annotation [94, 114, 332]. A selection strategy aims to identify instances that are likely to have the greatest impact on the classification performance of a model when used for re-training [332]. Our investigation focuses on computational-aware selection strategies that are either very fast during the Active Learning process or can be computed prior to training, thereby minimizing the overall duration of each Active Learning iteration. We concentrate on the following leigtweight acquisition functions:

- **Random**. A common baseline for Active Learning is to randomly select the next instance to be manually labeled from the pool of unlabeled examples [221]. This is also referred to as *Passive Learning*.
- **Uncertainty**. Uncertainty-based selection strategies [220] select data instances based on where the model is most uncertain about its label. We apply *smallest margin* sampling (Eq. 2.33) to quantify the uncertainty of predictions, referred to as $U(x)$. In a preliminary experiment, we identified smallest margin sampling as the most accurate and precise sampling strategy in our evaluation setting.
- **Density*Uncertainty**. Since highly uncertain instances may not be representative, Zhu et al. [407] suggest selecting instances according to the maximum uncertainty and the most representative in terms of density. The authors suggest a k-Nearest-Neighbor density measure to evaluate the density of an instance $i \in D_{pool}$ with respect to the K most similar examples, denoted as $S(x) = \{s_i\}_{i=1}^{K}$, which is defined as:

$$DS(x) = K^{-1} \sum_{s_i \in S(x)} \cos(x, s_i) \tag{7.2}$$

where $\cos(x, s_i)$ is the cosine-similarity between x and s_i. The score function, which accounts for uncertainty and density, is defined as:

$$DSH(x) = DS(x) \times U(x) \tag{7.3}$$

- **Instability**. Zhu and Ma [406] suggest a selection strategy based on instability. The instability of a prediction is measured by the changes in the prediction uncertainty scores over the last n consecutive learning cycles. The authors provide two instability measures, namely *Label-insensitive Instability Sampling* ($\mathrm{IS_{LI}}$) and *Label-sensitive Instability Sampling* ($\mathrm{IS_{LS}}$). $\mathrm{IS_{LS}}$ reports high uncertainties when instances cause unstable uncertainty estimates during the last l consecutive learning cycles, that is:

$$IS_{LI}(x) = U^i(x) + \sum_{i-l<k\leq i} U^k(x) - U^{k-1}(x) \tag{7.4}$$

where $U^i(x)$ denotes to the uncertainty of x at the ith learning iteration. $\mathrm{IS_{IS}}$ selects the most informative example from the set of unlabeled examples that have a high instability and different label predictions during the last consecutive learning cycles, that is:

$$IS_{LS}(x) = U^i(x) + \sum_{i-l<k\leq i} \mathbb{I}(y^k \neq y^{k-1}) \times \left(U^k(x) - U^{k-1}(x)\right) \tag{7.5}$$

- **Density*Instability**. Zhu and Ma [406] suggest selecting instances according to the maximum instability and highest density, which can be formulated as:

$$DSIS_{LI}(x) = DS(x) \times IS_{LI}(x) \tag{7.6}$$

$$DSIS_{LS}(x) = DS(x) \times IS_{LS}(x) \tag{7.7}$$

7.4 Study Design

This section outlines the design of our study. First, it states the research questions and describes the benchmark criteria. It then presents the dataset used and the implementation details.

7.4.1 Research Questions

We investigate the following research questions:

RQ1: How accurate is BERT when trained via Proxy-based Active Learning?

RQ1 focuses on evaluating the classification performance of a BERT Consumer trained via Proxy-based Active Learning. To identify the best-performing setup, we compare the suitability in terms of classification performance achieved by various lightweight Proxies and selection strategies.

RQ2: What is the gain in classification performance between the Proxy and the Consumer in our Proxy-based Active Learning framework?

RQ2 investigates whether the additional effort of using two models (a Proxy and a Consumer) during the learning process in Proxy-based Active Learning is profitable and worth the additional computational effort in terms of increased classification performance. As the Proxy, we consider a lightweight ML model to maintain low model latencies, while the Consumer is much more complex.

RQ3: How suitable is Proxy-based Active Learning for real-time labeling?

RQ3 examines the runtime behavior of Proxy-based Active Learning. This includes the process of training, inference, and data selection of the Proxy and whether it satisfies strict time requirements.

RQ4: How do uncertainty-based and random selection strategies differ in terms of the quality of the selected training data?

Previous research shows that uncertainty sampling for Active Learning leads to higher classification performance than random selection strategies [220]. It remains unknown how uncertainty sampling via a Proxy affects the quality of the training dataset compared to a random selection. This research question investigates the quality of the sampled training data.

7.4.2 Benchmark Criteria

To address RQ1, we first examine the classification performance of Proxy-based Active Learning using FastText and LR as the Proxy, with BERT serving as the Consumer. We report the mean of five independent model runs for each experiment, based on stratified train-test splits. Additionally, we assume the availability of a pool of unlabeled data instances and a small budget for labeling the initial training dataset. We evaluate the classification performance of the BERT Consumer using various uncertainty-based selection strategies and compare their performance against random selection. We consider a maximum labeling budget of 500.

To address RQ2, we explore the classification performance enhancements of our Proxy-based Active Learning framework, using BERT as the Consumer and

FastText and LR as the Proxy. We compare this approach against traditional Active Learning, where the same lightweight classifier is used for both data selection and deployment. We do not consider using BERT in Active Learning due to the impracticality caused by the high latency of BERT. We evaluate the gains in classification performance when training an additional state-of-the-art Consumer and determine whether it is worth the effort.

Regarding RQ3, we examine the real-time capability of our Proxy-based Active Learning framework to assess its applicability in real-world settings. To achieve this, we measure the time required to perform a learning iteration that includes model training, inference, and selection of the next most informative instance. We follow the rule of thumb suggesting that a user in an interactive setting should not wait more than 1 second for the response of his or her action to maintain user experience and productivity [248, 361].

For RQ4, we investigate the quality of the Proxy- and randomly sampled training data. As a measure of the quality of the training data, we consider the class balance, which is an essential property of high-quality training data [300, 395]. Class balance is maintained when no single class significantly outweighs the others. Typically, most ML models exhibit a bias toward the majority class, which leads to more frequent misclassification of the minority class. We use Shannon's entropy as a measure of imbalance. Furthermore, we assess the span of class ratios by outlining the maximum and minimum class ratios.

7.4.3 Datasets

We consider three publicly available English datasets that originate from real-world domains. The datasets are summarized in Table 7.1. The datasets exhibit imbalanced class distributions, a typical characteristic of real-world data. Generally, class imbalances present additional hurdles for both automated text classification and Active Learning [131, 395], including biases toward selecting the majority class. Nonetheless, imbalanced classes offer a more realistic evaluation scenario.

Dataset	Size	$\lvert C \rvert$	Class Distribution	#Words ($\mu \pm \sigma$)
App Store	6,392	3	3,855:1,437:1,100	24 ± 29
Hate Speech	24,783	2	20,620:4,163	14 ± 7
Reuters	8,759	9	3,930:2,319:527:495: 458:425:282:166:157	152 ± 176

Table 7.1: Statistics about the datasets used, include its size, the number of classes and their distribution, as well as the mean and standard deviation of words per text instance.

First, we utilize the **App Store** dataset from the domain of participatory requirements engineering [239]. This dataset consists of manually labeled app reviews covering *feature requests*, *bug reports*, and *praise*. Second, we employ the **Hate Speech** dataset [80], which contains tweets manually labeled for toxicity (*toxic* and *non-toxic*). Third, we use the **Reuters** dataset [222], which consists of a highly imbalanced topic modeling task. For our experiments, we select a subset of the nine most frequent topics with unambiguous labels. We partition all datasets into a 50% training set (data pool) and a 50% test set, maintaining the original label distribution. To train the BERT classifier, we allocate 10% of the selected training data as a validation set.

7.4.4 Implementation Details

We apply the following configuration for Proxy-based Active Learning: for the App Store and Hate Speech datasets we randomly sample 10 instances per class as the initial training data. For the Reuters dataset, we randomly sample 3 instances per class, resulting in an initial training dataset of 30, 20, and 27 instances, respectively. In each iteration, only one instance is selected from the pool and added to the training dataset. Training is always performed from scratch. In total, we run 500 training iterations, causing a labeling budget of 500 instances. We perform our experiments on fully labeled datasets, which allows us to simulate manual labeling. This approach is consistent with the standard method for evaluating the classification performance of Active Learning approaches [94, 114, 221].

For the LR classifier, we utilize the default implementation provided by the Scikit-learn library [286], with the maximum number of iterations set to 100. We employ the FastText implementation provided by Joulin et al. [175], setting the embedding size to 10, training for 5 epochs, and employing a learning rate of 0.1. For the FastText classifier, we do not leverage the density of the learned sentence representation for the query strategy because this information is not available in their FastText implementation. Our BERT implementation relies on Hugging Face [390]. We use the *bert-base-uncased* pre-trained model and perform fine-tuning over 5 iterations. In our experiments, we set $K = 20$ to estimate the density (Eq. 7.2) as proposed by the original authors. Furthermore, we compute the instability (Eq. 7.4 and 7.5) over the last $l = 5$ iterations. All experiments are performed on an Intel® Core™ i7-8550U CPU @ 1.80GHz with 16 GB of main memory.

Strategy	App Store				Hate Speech				Reuters					
	FastText	LR+ SBERT	LR+ BERT	LR+ ELMo	FastText	LR+ SBERT	LR+ BERT	LR+ ELMo	FastText	LR+ SBERT	LR+ BERT	LR+ ELMo		
Random	83.30	83.07	83.03	83.12	88.29	87.91	87.93	88.51	88.38	88.28	89.11	88.85	micro F1 score	300 Iterations
U	84.42	83.24	84.69	83.78	91.24	88.27	90.94	**93.02**	**91.33**	89.69	**92.15**	**91.43**		
U*DS	-	83.48	**84.59**	84.32	-	91.16	90.90	**92.27**	-	89.44	91.39	**92.33**		
IS$_{LI}$	83.15	82.49	84.02	83.26	91.13	89.36	**91.43**	92.03	86.30	91.11	**92.23**	90.59		
IS$_{LI}$*DS	-	83.50	**85.03**	81.93	-	87.64	91.50	91.87	-	90.45	**92.42**	**92.62**		
IS$_{LS}$	83.92	82.27	**85.13**	83.59	90.52	85.28	**91.85**	**92.45**	86.88	**90.98**	**92.82**	91.14		
IS$_{LS}$*DS	-	82.43	**84.95**	82.57	-	90.12	91.07	**92.56**	-	90.49	**93.16**	**91.54**		
AVG (unc)	83.83	82.78	84.74	83.24	90.96	88.64	91.28	92.37	88.18	90.36	92.36	91.61		
Random	78.50	78.18	78.19	78.27	70.36	69.07	69.25	71.22	65.79	66.16	68.67	67.17	macro F1 score	
U	79.98	78.87	80.62	78.78	83.76	80.34	82.87	87.20	74.61	**80.25**	**79.99**	**79.25**		
U*DS	-	79.05	80.47	79.73	-	84.82	83.46	85.30	-	**80.36**	**79.06**	**79.71**		
IS$_{LI}$	78.14	77.55	79.43	78.55	84.21	80.65	83.77	85.10	63.00	**83.62**	**79.02**	72.37		
IS$_{LI}$*DS	-	79.11	**81.08**	77.22	-	78.17	83.18	85.03	-	**83.25**	**80.48**	**79.74**		
IS$_{LS}$	79.35	77.42	**81.08**	79.51	83.16	76.91	84.29	86.32	67.32	**82.96**	**79.56**	**79.52**		
IS$_{LS}$*DS	-	78.05	**80.84**	77.95	-	82.49	83.03	86.60	-	82.41	**80.28**	**80.00**		
AVG (unc)	79.16	78.12	80.59	78.62	83.71	80.56	83.44	85.93	68.31	82.14	79.73	78.43		
Random	85.06	84.64	85.06	84.76	93.09	93.09	93.23	92.84	92.54	92.29	92.56	92.50	micro F1 score	500 Iterations
U	**86.29**	84.71	**86.55**	**86.23**	93.83	93.39	93.60	**94.35**	**94.56**	**95.41**	**95.38**	**95.19**		
U*DS	-	85.04	**86.38**	**86.08**	-	93.49	**94.30**	**94.62**	-	**95.39**	**95.60**	**95.44**		
IS$_{LI}$	85.96	85.27	**86.38**	**86.31**	**94.31**	92.77	**94.29**	**94.54**	91.21	**95.12**	**95.33**	**95.95**		
IS$_{LI}$*DS	-	85.47	86.27	85.06	-	93.58	**94.28**	**94.39**	-	**94.97**	**95.37**	**95.08**		
IS$_{LS}$	85.96	84.99	**86.23**	85.64	**94.29**	93.58	**93.98**	**94.51**	92.52	**95.13**	**94.90**	**95.65**		
IS$_{LS}$*DS	-	85.61	**86.45**	85.66	-	**93.98**	**93.94**	**94.64**	-	**95.37**	**95.32**	**95.37**		
AVG (unc)	86.07	85.18	86.38	85.83	94.14	93.47	94.06	94.51	92.76	95.23	95.32	95.45		
Random	80.75	80.32	80.84	80.48	86.92	87.09	87.46	86.64	76.57	75.40	76.91	75.75	macro F1 score	
U	82.48	80.49	**82.89**	**82.41**	88.93	88.55	88.35	**89.95**	82.50	**91.32**	**87.18**	**87.01**		
U*DS	-	81.18	**82.59**	**82.11**	-	88.51	**89.61**	**90.25**	-	**91.04**	**88.68**	**88.78**		
IS$_{LI}$	82.27	81.32	**82.71**	**82.66**	**89.71**	87.45	**89.63**	**90.25**	77.49	**90.69**	**88.42**	**91.15**		
IS$_{LI}$*DS	-	81.66	82.29	80.87	-	88.95	**89.75**	**90.14**	-	**90.45**	**88.01**	**87.22**		
IS$_{LS}$	82.35	81.13	82.49	81.83	**89.87**	88.62	**89.12**	**90.08**	82.48	**90.31**	**84.80**	**88.40**		
IS$_{LS}$*DS	-	81.99	**82.77**	81.61	-	89.55	88.57	**90.65**	-	**91.42**	**87.55**	**87.35**		
AVG (unc)	82.37	81.30	82.62	81.92	89.50	88.61	89.17	90.22	80.82	90.87	87.44	88.32		

Table 7.2: Micro and macro F1 scores of the BERT Consumer after 300 and 500 iterations. The best and worst performing selection strategies for each classifier are highlighted in green and red, respectively.

7.5 Results

We present the results of our experiments and answer the four research questions. The source code of our experiments is publicly available[1].

7.5.1 Classification Performance

Table 7.2 lists the F1 scores of BERT Consumers when trained using Proxy-based Active Learning for 300 and 500 iterations, respectively. The table is organized by sampling strategies, Proxies, datasets, and number of Active Learning iterations. The average (AVG (unc)) F1 scores across all uncertainty-based selection strategies are provided at the bottom of each table. Significant improvements (paired t-test with p-value < 0.05) compared to a random selection strategy are highlighted in bold. The best and worst-performing embeddings for each dataset and selection strategy are highlighted in green and red, respectively.

Our results show that, across all datasets, Proxy-based Active Learning can significantly improve the F1 score of a BERT classifier (Proxy) compared to training BERT on randomly sampled training data. After 500 iterations, a

[1] `https://github.com/jsandersen/ProxyAL`

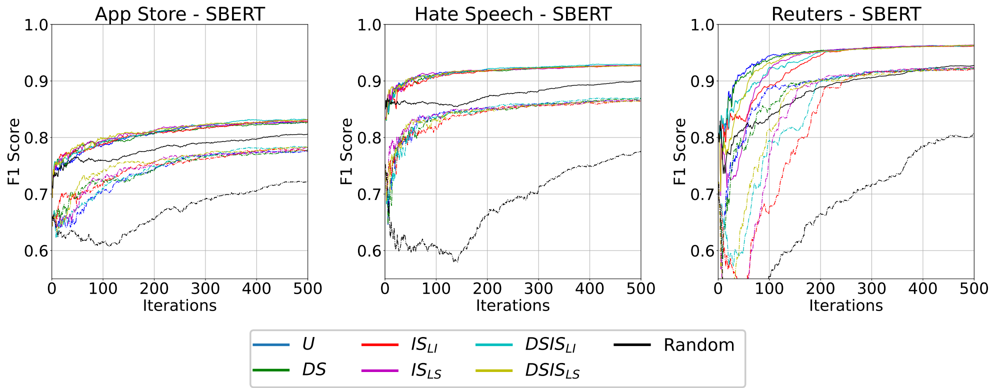


Figure 7.3: Learning curves of the traditional Active Learning process with SBERT. Solid and dashed lines represent the macro and micro F1 scores, respectively.

relative improvement in the micro F1 score of 1.75%, 1.15%, and 3.18% was achieved for each dataset, respectively. The macro F1 score increased by 2.54%, 2.62%, and 15.30%, respectively. Across all experiments, the BERT and ELMo embeddings provide the best Consumer performance. FastText provides the second-best results on the App Store dataset, but does not perform well on the Reuters dataset. The SBERT embeddings perform the worst, achieving significant improvements only on the Reuters dataset. The F1 score improvements between the 300th and 500th iterations are straightforward, as more instances are used for training. Also, the macro F1 score improvements were much higher than the micro F1 score improvements due to the highly imbalanced datasets. None of the strategies consistently outperformed the others, a common effect when evaluating Active Learning [94] pipelines. Overall, the results show that Proxy-based Active Learning can significantly improve the F1 score compared to randomly sampled instances, which would cause no latency.

7.5.2 Comparison with Traditional Active Learning

The question is whether it is worthwhile to train an additional BERT classifier, as done in Proxy-based Active Learning, or whether a traditional Active Learning approach (using a lightweight classifier) would yield similar F1 scores. Figure 7.3 illustrates the relative improvement of a FastText and LR classifier used in traditional Active Learning compared to training an additional BERT classifier on the selected training set.

The results indicate that consecutive training of BERT Consumers can improve the micro F1 score by up to 7.27% and the macro F1 score by up to 19.34% compared to deploying the Proxy after the traditional Active Learning

Strategy	App Store FastText	App Store LR+ SBERT	App Store LR+ BERT	App Store LR+ ELMo	Hate Speech FastText	Hate Speech LR+ SBERT	Hate Speech LR+ BERT	Hate Speech LR+ ELMo	Reuters FastText	Reuters LR+ SBERT	Reuters LR+ BERT	Reuters LR+ ELMo		
U	1.34	1.37	5.31	1.97	3.34	-4.43	3.21	6.24	3.34	-6.50	1.62	-2.27	micro F1 score	300 Iterations
U*DS	-	2.19	5.70	2.01	-	-1.23	2.94	5.56	-	-6.75	0.76	-1.24		
IS_{LI}	-0.17	0.99	4.63	1.26	3.21	-3.15	4.14	4.89	-2.36	-5.00	2.53	-3.06		
IS_{LI}*DS	-	1.72	5.78	-0.37	-	-5.19	3.92	4.67	-	-5.64	2.88	-0.76		
IS_{LS}	0.74	0.14	6.57	1.17	2.53	-7.48	4.36	5.31	-1.70	-5.08	2.46	-2.65		
IS_{LS}*DS	-	-0.82	6.25	0.73	-	-2.34	3.40	5.57	-	-5.39	3.18	-2.01		
AVG (unc)	0.57	0.93	5.71	1.13	3.03	-3.97	3.66	5.37	-0.24	-5.73	2.24	-2.00		
U	1.88	2.82	8.25	1.73	19.05	-6.54	8.72	17.99	13.42	-12.44	-2.70	-9.24	macro F1 score	
U*DS	-	3.28	8.64	1.83	-	-1.13	8.21	16.23	-	-12.23	-3.96	-9.14		
IS_{LI}	-0.45	1.79	7.50	1.93	19.69	-5.83	10.14	12.66	-4.23	-8.56	2.06	-16.59		
IS_{LI}*DS	-	3.15	9.61	0.05	-	-9.05	8.30	13.15	-	-9.12	0.92	-8.74		
IS_{LS}	1.08	0.53	10.70	2.08	18.19	-10.33	10.80	15.28	2.33	-9.47	-2.68	-9.14		
IS_{LS}*DS	-	-0.70	10.01	1.42	-	-3.86	9.32	15.25	-	-9.49	-1.32	-8.09		
AVG (unc)	0.84	1.81	9.12	1.51	18.98	-6.12	9.25	15.09	3.84	-10.22	-1.97	-10.16		
U	1.45	2.44	6.59	3.66	0.80	0.72	5.55	6.82	2.19	-0.84	3.66	0.90	micro F1 score	500 Iterations
U*DS	-	2.81	6.73	3.67	-	0.76	6.06	7.27	-	-0.89	4.03	1.22		
IS_{LI}	1.03	2.89	6.58	3.52	1.31	0.15	6.49	6.86	-1.43	-1.10	3.66	1.73		
IS_{LI}*DS	-	2.79	7.00	2.01	-	0.68	6.30	6.74	-	-1.34	3.96	0.85		
IS_{LS}	1.07	2.33	6.12	2.71	1.29	0.93	6.09	7.26	-0.02	-1.29	3.25	1.40		
IS_{LS} *DS	-	2.95	6.76	2.79	-	1.29	5.95	6.95	-	-0.99	3.83	1.11		
AVG (unc)	1.18	2.70	6.63	3.06	1.13	0.75	6.07	6.98	0.24	-1.08	3.73	1.20		
U	2.14	3.80	9.80	4.63	2.32	2.36	13.82	18.61	7.75	-0.90	2.54	-2.22	macro F1 score	
U*DS	-	4.37	9.93	4.49	-	1.95	14.44	19.34	-	-1.27	5.05	-0.39		
IS_{LI}	1.88	4.52	10.11	4.95	3.22	1.25	15.57	16.71	1.21	-1.50	3.74	2.54		
IS_{LI}*DS	-	4.29	10.73	2.55	-	2.31	15.30	17.20	-	-1.99	4.53	-2.06		
IS_{LS}	1.98	3.77	9.81	3.67	3.40	2.41	15.43	18.73	7.72	-2.28	0.10	-0.66		
IS_{LS}*DS	-	4.70	10.29	3.95	-	3.40	14.85	17.84	-	-0.98	3.91	-1.98		
AVG (unc)	2.00	4.24	10.11	4.04	2.98	2.29	14.90	18.07	5.56	-1.49	3.31	-0.80		

Table 7.3: Relative F1 score improvements of a BERT Consumer compared to FastText and LR as the Proxy using 300 and 500 iterations. The best and worst performing selection strategies for each classifier are highlighted in green and red, respectively.

process. However, the FastText and LR Proxies are very strong baselines on their own. Using less training data, i.e., 300 instances, only BERT, ELMo, and sometimes FastText provide strong improvements (> 3%), while SBERT performs similarly or even worse than the Proxy alone. In contrast, using 500 instances as the training data for Proxy-based Active Learning, the App Store, and the Hate Speech datasets show strong improvements. Only the Reuters dataset exhibits no improvement over a stand-alone Proxy. Proxy-based Active Learning benefits from a larger number of iterations, as all F1 scores improve when 500 iterations are performed instead of 300. While SBERT embeddings achieve the highest F1 scores when applying self-selection, they provide the worst improvements within Proxy-based Active Learning.

7.5.3 Run-time Investigation

Next, we investigate the time behavior of Proxy-based Active Learning. Table 7.4 displays the time needed to perform the 500th learning iteration for each Proxy. The table shows the average run-time of all six selection strategies. Averaging is performed to keep the figure clear since no large differences between the selection strategies were observed. We perform all experiments on a CPU as outlined in Section 7.4.4.

Runtime	App Store			
	FastText	LR+SBERT	LR+BERT	LR+ELMo
Training	6.62	0.08	0.17	0.20
Inference	0.11	0.01	0.01	0.01
Selection	0.08	0.12	0.12	0.15
Total	6.81	0.21	0.30	0.36

Runtime	Hate Speech			
	FastText	LR+SBERT	LR+BERT	LR+ELMo
Training	6.68	0.14	0.27	0.30
Inference	0.31	0.02	0.01	0.02
Selection	0.29	0.14	0.13	0.16
Total	7.28	0.29	0.41	0.49

Runtime	Reuters			
	FastText	LR+SBERT	LR+BERT	LR+ELMo
Training	7.42	0.04	0.14	0.15
Inference	0.53	0.04	0.02	0.05
Selection	0.37	0.39	0.39	0.48
Total	8.32	0.47	0.54	0.68

Table 7.4: Run-time of the 500th Active Learning iteration in seconds on a CPU using FastText and LR.

All steps in the Active Learning loop (training, inference, and data selection) were carried out in less than 1 second using LR with pre-trained text encodings. SBERT is the fastest approach, followed by BERT and ELMo, which took up to 0.68 seconds for the 500th iteration. FastText is much slower, taking > 6 seconds for the 500th iteration, which is too slow for real-time Active Learning. Overall, the total runtime grows linearly with the number of iterations (within the first 500 iterations). Furthermore, the runtimes indicate that a batch size of one is appropriate and that there is no need to use batch-based selection strategies to save runtime even on a CPU.

7.5.4 Sampled Dataset Quality

Lastly, we investigate the imbalance of the training data sampled by a Proxy compared to random selection (Passive Learning). A balanced dataset is one where the number of instances in each class is roughly equal or follows a similar distribution. On the other hand, an imbalanced dataset is one in which the number of instances in different classes varies significantly, with one or more classes having a much smaller number of instances than others. Imbalanced datasets can pose challenges for ML models, particularly when the minority class (the class with fewer instances) is of interest and needs to be accurately predicted. In such cases, the ML model may have a bias toward the majority class, leading to poor performance in predicting the minority class. Table 7.5 presents metrics evaluating the imbalance of the sampled data, including Shannon's entropy (Entropy) and the maximum and minimum class ratios.

Metrics	Proxy-based Data Sampling				Random	App Store
	FastText	LR+SBERT	LR+BERT	LR+ELMo		
Imbalance (Entropy)	0.962	0.981	**0.989**	0.984	0.877	
Max Class Ratio	0.43	0.40	0.40	0.42	0.58	
Min Class Ratio	0.23	0.24	0.28	0.28	0.18	

Metrics	Proxy-based Data Sampling				Random	Hate Speech
	FastText	LR+SBERT	LR+BERT	LR+ELMo		
Imbalance (Entropy)	0.997	**1.000**	0.995	0.992	0.689	
Max Class Ratio	0.53	0.51	0.54	0.55	0.82	
Min Class Ratio	0.47	0.49	0.46	0.45	0.18	

Metrics	Proxy-based Data Sampling				Random	Reuters
	FastText	LR+SBERT	LR+BERT	LR+ELMo		
Imbalance (Entropy)	0.876	**0.959**	0.941	0.944	0.753	
Max Class Ratio	0.31	0.20	0.22	0.21	0.43	
Min Class Ratio	0.02	0.06	0.04	0.04	0.02	

Table 7.5: Quality indicators of the Proxy-sampled datasets.

Our results show that random sampling approximates the class distribution of the data pool. For example, in the binary Hate Speech dataset, after 500 iterations, 82% of the training instances are from the same class. Our experiments demonstrate that Proxy sampling is much less prone to sample disproportionately from majority classes. We found that the Proxy-sampled data is highly balanced. The best balance is observed on the Hate Speech dataset with an entropy greater than 0.99, which is a highly precise approximation of a perfect 50:50 balance. In the App Store dataset, a 40:24:36 split is observed. Figure 7.4 illustrates the concrete ratio of classes throughout the training process for better visualization. Each class is represented by a different color. The class ratio of the 500th iteration aligns with the values in Table 7.5. Our results show that Proxy sampling effectively reduces the model's bias toward a particular class by providing much more balanced training data.

7.6 Discussion

We outline the implications of our findings, explore the range of domains in which Proxy-based Active Learning can be applied, highlight limitations, and review related studies on alternative approaches to improve the applicability of Active Learning in real-world settings.

7.6.1 Implications

Our benchmark results demonstrate that Proxy-based Active Learning can significantly improve the F1 score of text classifiers in a low-budget labeling setting.

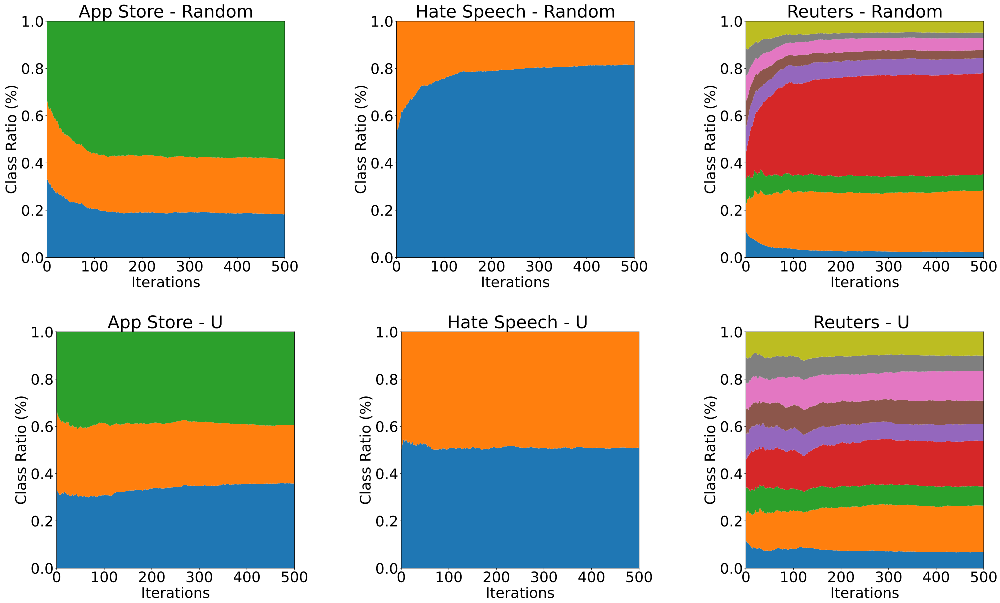


Figure 7.4: Class ratio of the training dataset sampled randomly compared to a Proxy using SBERT encodings and an LR classifier.

We found that our framework increases the micro F1 score of a BERT Consumer by up to 7.27% and the macro F1 score by up to 19.34% compared to deploying the lightweight Proxy. We considered a manual labeling budget of only 500 instances. Furthermore, we showed that a BERT model trained via Proxy-based Active Learning increases the micro and macro F1 scores by up to 3.18% and 15.30%, respectively, compared to training BERT with a randomly sampled dataset. Additionally, we show that Proxy-based Active Learning requires far less than 1 second for each iteration. This finding is of great importance, as time constraints are a major obstacle to the applicability of HiL systems. In particular, we demonstrate that an LR classifier is up to ~ 33 times faster and provides better reusability of the sampled data than FastText, previously considered the state-of-the-art for rapid foreign selection [71]. Overall, the most critical factors for the effectiveness of Proxy-based Active Learning compared to Active Learning are that this approach does not increase the data labeling effort, provides significantly better results than Passive Learning, and allows for fast interaction cycles.

Our results also indicate that Proxy-based Active Learning does not require many data instances to learn. The difference between labeling 300 and 500 instances results in an average F1 score growth rate of 3.6%. Furthermore, Proxy-based Active Learning requires a very small amount of initial labeling data, ranging from 3 to 10 instances per class, while supporting a batch size of one. By comparison, some Active Learning studies require extraordinary human effort, which is impossible to offer in most real-world settings. For example, Hu et al. [159] consider an initial training dataset and batch sizes of 500 to 2,500 instances and a total labeling budget of 10,000 to 25,000 instances. Most Active Learning studies use a batch size between 20 and 100 instances [125, 154].

F1 score	**App Store**	**Hate Speech**	**Reuters**
micro	87.82	96.19	97.17
macro	84.30	93.21	93.94

Table 7.6: The maximum reachable F1 score that can be achieved when the entire data pool is used to train BERT.

In addition, the classification performance of the Consumer was close to the maximum reachable F1 score of BERT, as shown in Table 7.6. The depicted F1 scores are obtained when the entire data pool is labeled and used for training. The table shows that on the Reuters dataset, the Proxy already reaches up to 99% of the maximum reachable F1 score after 500 iterations. In this case, training an additional BERT classifier does not provide any improvement over the original Proxy. In comparison, the Proxy reaches only 90–96% of the maximum F1 score on the App Store and Hate Speech datasets. Thus, Proxy-based Active Learning is most beneficial when the maximum reachable F1 score of the Proxy is much lower than that of the Consumer. Otherwise, it may be sufficient to directly deploy the lightweight Proxy to save computational effort.

7.6.2 Field of Application

The use-case for Proxy-based Active Learning is to maintain a high user experience while reducing the cost of training well-performing state-of-the-art text classifiers. It aims to take advantage of Active Learning when the desired target model is actually too slow to provide fast HiL interactions. Like Active Learning, Proxy-based Active Learning is domain-independent and can be applied to any classifier that provides uncertainty estimates along the classification result.

We demonstrate that the data labeling phase of Proxy-based Active Learning can be performed in near real-time on low-end infrastructure. This enables the integration of Active Learning-like training into ML tools without compromising user experience due to long waiting times and interruptions. In addition to fastening existing Active Learning-based labeling tools, one area of application is data analysis and exploration tools such as Forum 4.0 [134]. Such tools typically allow domain experts to dynamically build classifiers from scratch. A lightweight and real-time learning framework, such as Proxy-based Active Learning, allows users to rapidly develop, test, and deploy ML models while maintaining a high level of user experience in terms of latency. A typical example is making sense of user comments from marketplaces or social media channels.

A second field of application is to simplify the development and deployment of classifiers. The Proxy can be developed quickly even by domain experts using their personal workstations. The speed also allows for rapid testing and prototyping of classification models before committing to resource-intensive computations, such as fine-tuning a full BERT model. The speed of Proxy-based Active Learning is particularly useful for rapid prototyping, i.e. when the classification objective is vague and not fully developed at the outset. Initial evaluation results from the Proxy could give an indication of how well a particular task can be solved.

Finally, our results show that existing Active Learning applications are likely to benefit from additional training of a state-of-the-art Consumer on the already acquired training data. Thus, training new and more powerful ML models on existing data has the potential to lead to further improvements in classification performance. However, our study only covers the transfer of training data from traditional classifiers to BERT.

7.6.3 Threats to Validity

We discuss threads to the internal and external validity.

Internal validity. Our benchmark experiments cover a limited number of classification models (Proxy and Consumer) and sampling strategies. Although we used classifiers that have been shown to perform well in text classification, we may have missed the optimal configuration or parametrization of Proxy-based Active Learning that could improve the results.

Furthermore, we assume that the human labelers provide flawless ground truth labels, which may be unrealistic. Humans are also error-prone, and some mislabeling is to be expected. Noise in the training data can corrupt the classification performance of the trained classifier. However, in the research domain of Active Learning, it is an established approach to consider the provided labels as correct [94, 154, 221].

In addition, our experiments are based on human labeled datasets. As humans can be subject to bias or error, noise may have been introduced into the data, distorting the labeling results. To mitigate this problem, we relied on established benchmark datasets from the literature.

External validity. The characteristics of the dataset used can affect the validity of the results. In particular, the distribution and quality of the data, i.e. the presence of outliers or anomalies, can affect the classification performance. Generalizing the results may be difficult if the dataset used is not representative or tends to change over time.

Furthermore, in our experiments we used an additional hold-out dataset to reliably estimate the classification performance of the model during learning. In the real world, however, a hold-out dataset is typically not available, making it difficult to assess the classification performance of Active Moderation during deployment.

Proxy-based Active Learning aims to reduce the cost of developing an appropriate training dataset. However, we only measure the cost of labeling in terms of the number of labels assigned. However, the true cost of labeling also depends on the time a user has to spend on labeling. Assuming that labeling very long and very short text takes the same amount of time is insufficient, but common practice in Active Learning research. Human constraints, such as time and budget limitations, may restrict applicability of Active Learning.

7.6.4 Alternative Approaches from Related Work

The idea of using two ML models to train a deployable classifier was first discussed by Tomanek and Morik [362]. They raise the so-called re-usability problem of Active Learning, which is about whether *"a set of labeled examples that is deemed most informative using one classification algorithm necessarily informative for another classification algorithm?"*. They study the re-usability problem using traditional ML classifiers in the domain of text classification. They found that foreign selection is around 75% of the cases better than a random selection strategy. Hu et al. [160] investigate the mutual re-usability of pairs of traditional text classifiers. They explore which combination of Proxy and Consumer provides the best classification performance. In contrast to our work, they do not consider the time savings of less complex Proxies. Lowell et al. [235] examine Proxy-based data sampling between similarly accurate classifiers, including deep learning. We focus on the transferability between a fast classifier and a state-of-the-art deep learning approach. Coleman et al. [71] explore the time-savings and errors using a FastText classifier as the Proxy and a deep NN as the Consumer. They show that FastText is up to 41.9 times faster with no significant increase in errors and no loss of classification performance. However, their approach still takes multiple minutes to train, which is too slow for real-time processing. Prabhu et al. [293] also investigate Proxy-based Active Learning using FastText, but they rely on very large labeling budgets. In contrast, we focus on a low-budget real-time setting, where no more than 500 instances are queried during the Active Learning process.

Proxy-based Active Learning is also related to approaches that address the trade-off between classification performance and computational time of Active Learning. A common practice to fasten Active Learning is to select new instances in batches [154] since training the model for each instance can be too computationally expensive. Batch-based sampling strategies aim to reduce waiting times within batches by reducing the number of training cycles. However, they still require a full re-training between each batch. Another approach is knowledge distillation [399], which aims to reduce the complexity of classifiers while preserving their performance. However, even small NNs are usually too complex to be trained in real-time.

7.7 Conclusion

This chapter has introduced a HiL framework for classifier training called Proxy-based Active Learning. Our framework aims to exploit the substantial classification performance of state-of-the-art models while maintaining low model latency and high user experience during the traditional Active Learning loop, a key requirement of HiL systems. Proxy-based Active Learning is motivated by the increasing complexity and computational time required to train state-of-the-art ML models, making them unusable with Active Learning from the user's perspective. We have performed a series of ML benchmark experiments to demonstrate the sufficiency and applicability of our framework. Our main findings are summarized below.

- BERT can be effectively trained using our framework. Our experiments demonstrate that Proxy-based Active Learning can improve the macro F1 score by 19.34% compared to a lightweight classifier trained via Active Learning (Proxy).

- BERT trained with Proxy-based Active Learning achieves up to 15.30% higher macro F1 scores than BERT (Consumer) trained with Passive Learning.

- Lightweight Proxies can be trained time-efficiently in near real-time on low-end infrastructure. For example, an LR Proxy is very fast, taking less than 1 second for each Active Learning iteration.

- LR as a Proxy significantly outperforms FastText in both runtime efficiency and F1 score.
- Uncertainty sampling yields significantly higher quality training data than random sampling, especially in terms of class balance.

Chapter 8

Explainable Uncertainty Estimation for Text Classification

Publication. The uncertainty explanation framework described in this chapter was originally published in the 2020 paper "Word-Level Uncertainty Estimation for Black-Box Text Classifiers using RNNs" [12]. My primary areas of involvement include designing and implementing the explanation framework, conducting the experiments, analyzing the results, and writing the research paper. In addition, this chapter draws on the 2023 paper "Explaining Prediction Uncertainty in Text Classification: The DUX Approach" [13]. My contribution to this paper involves designing and conducting the human evaluation, analyzing and reporting the results, and leading the writing of the paper.

Contribution. This chapter introduces BayLUXT, a novel Bayesian framework for providing local explanations of text classification results. BayLUXT is a self-explanatory and model-specific extension of RNNs such as LSTMs. While previous local explanation techniques only provide deterministic word relevance or attribution scores, BayLUXT additionally explains how much uncertainty each word contributes or takes away from the final class result. Furthermore, BayLUXT is able to explain the sequential understanding of RNN-based text classifiers and their uncertainty. BayLUXT integrates uncertainty modeling into explanations via Bayesian approximation. Hereby, aleatory and epistemic uncertainties are made explicit at the word-level. We conduct an experimental and human evaluation of the capabilities of BayLUXT. We show that BayLUXT is capable of effectively decomposing prediction uncertainty and that these explanations provide valuable insights for humans to more effectively understand text classification outcomes. In our human evaluation, more than 92% of the respondents state that explaining uncertainty is a desirable extension to traditional relevance-based explanations to facilitate the understanding of text classification outcomes.

8.1 Motivation

Clearly, it is important for humans to understand how a text classifier arrives at its predictions in order to build trust [52]. Unfortunately, ML-based text classifiers are considerable back-boxes as they do not reveal why certain predictions were made. The field of XAI, as introduced in Section 4.4.2, is dedicated to enhancing the human understandability and transparency of ML models, including text classifiers. In particular, local explanations [27, 238, 309] have shown to offer valuable details and insights that make a model's decisions much more understandable and trustworthy. Accountable insights into the decision-making process of ML models allow for greater attention to be paid to particularly error-prone and unreliable predictions, and a better understanding of why wrong decisions occur. Explanations are a core mechanism for HiL to facilitate collaboration between humans and ML models.

Current XAI techniques for text classification provide insight into which words contribute most to a class outcome [309]. These explain why a classifier favored one predicted class label over the others. However, just as models can fail by producing incorrect classifications, explanations can also be misleading and promote incorrect predictions. In particular, uncertainties can corrupt classification results, resulting in highly ambiguous and distorted explanations. A major obstacle to conventional XAI explanations is their persuasiveness [30]. Humans tend to be easily persuaded by explanations that actually promote false or highly uncertain results.

While previous local explanation techniques attempt to explain the words that contribute most to the final class label, they do not provide insight into which particular word contributed to the uncertainty of the classification [27, 238, 309]. Human users cannot be made explicitly aware of uncertainties that may corrupt predictions. Uncertainty estimation is another black-box component that provides opaque uncertainty scores. It remains unknown why a single prediction is considered uncertain, or why it is much more uncertain than others. This lack of understanding makes it challenging to acknowledge potential classification difficulties and errors.

The previous chapter illustrated the importance of recognizing when a model lacks the knowledge to produce reliable results. This understanding empowers humans to approach error-prone and unreliable predictions with additional caution, such as through Active Moderation. This lack of clarity poses a significant challenge, especially since many HiL deployment patterns rely on humans to deal with the most uncertain instances. We argue that explaining prediction uncertainty would provide a deeper understanding of model behavior.

8.2 Conceptual Framework

This section is dedicated to the BayLUXT framework. First, it outlines the problem to be solved. Then it describes the inner workings of BayLUXT. Finally, we illustrate how the word attributions are visualized.

8.2.1 Problem Statement

Let f^{ω} be the prediction function of a trained NN classifier with its learned parameters $\omega \in \Omega$. Given an input sequence of word embeddings[1] $e = (\epsilon_1, \epsilon_2, ..., \epsilon_E) \in X$ with dimensionality E, f^{ω} computes class activation score S_c^{ω} per class $c \in Y$ given e. The class activation score S_c^{ω} is the outcome of the last layer before the final activation function ϕ. The class prediction y is the class with the highest activation score that is,

$$y = f^{\omega}(e) := \arg\max_{c \in Y} S_c^{\omega}(e) \tag{8.1}$$

The posterior probability of a class prediction is computed by applying a softmax function ϕ to the class activation scores:

$$p(y = c|e, \omega) := \phi(S_c^{\omega}(e)) = \frac{\exp(S_c^{\omega}(e))}{\sum_{k \in Y} \exp(S_k^{\omega}(e))} \tag{8.2}$$

Given some $e \in X$, a mapping $\exp : X \rightarrow \mathbb{R}^E$ is desired that explains a prediction pair $(e, y) \in X \times Y$. Traditionally, an explanation should capture how much each word of the input text e contributes to the final class prediction y. We aim to compute a mapping that explains the predicted uncertainty $u[y|e]$ regarding the influence of each word ϵ in e. It is expected that some words will add uncertainty, while others will remove it, analogous to word relevance. To further improve the understanding of how text classifiers work, we aim to assess the parts of the input that a model can classify easily or has problems with. We refer to this as explaining the sequential behavior of a model. The following explanations should be provided by BayLUXT:

- Explain the contribution of each word ϵ in e to the posterior probability $p(y = c|e, D)$ for all classes $c \in Y$.
- Explain the contribution of each word ϵ in e to the prediction uncertainty $u[y = c|e]$ for all classes $c \in Y$.

[1] We refer to a text instance as e rather than x to emphasize that the input must explicitly be a sequence of word embeddings. Thus, in this chapter, $e \in X \subset \mathbb{N}^{E \times d}$ applies. Where d is the dimensionality of the word embedding.

- Explain the posterior probability $p(y = c|(\epsilon_1, \epsilon_2, ..., \epsilon_i), D)$ for each index $1 \leq i \leq E$ and for all classes $c \in Y$.
- Explain the prediction uncertainty $u[y = c|(\epsilon_1, \epsilon_2, ..., \epsilon_i)]$ for each index $1 \leq i \leq E$ and for all classes $c \in Y$.

In addition, the estimated uncertainty of the prediction should be separable into its aleatory and epistemic components.

8.2.2 Uncertainty Modeling

To explain prediction uncertainties, they must first be modeled and quantified. For our implementation of BayLUXT, we consider MCD to model uncertainties. By enabling dropout at inference time, each forward pass uses a random sample of weights, resulting in a probabilistic model. A measure of prediction uncertainty with respect to an input e is derived by analyzing the statistical dispersion of the output distribution $p(y = c|e, D)$. For BayLUXT we apply the uncertainty quantification approach proposed by Kwon et al. (Eq. 2.38) to estimate the total uncertainty U and decompose it into its epistemic and aleatory components. Since their approach requires no structural changes, uncertainty modeling can be seamlessly integrated into existing RNN implementations.

8.2.3 Decomposition of Word Relevance and Uncertainty using RNNs

BayLUXT follows a model-specific sequence attribution approach using RNNs in conjunction with Bayesian modeling to assess word and sequence uncertainty. Explanations are modeled as an additional outcome of the inference process. An RNN, as introduced in Section 2.1.6, is an adaptation of a traditional FFNN for processing sequence data. Each element of the input sequence is recurrently evaluated, taking into account information from the previously processed sequence. We consider an RNN variant called Long Short Term Memory (LSTM), initially described by Hochreiter and Schmidhuber [152]. LSTMs are commonly used for text classification tasks [223, 276, 330] as they perform much better than traditional RNNs. We briefly describe how we derive relevance and uncertainty explanations using an LSTM.

An LSTM consists of a sequence of repeated cells, each of which computes a hidden state. For each index t of the input sequence, the output of a corresponding cell is controlled by a set of gates as a function of an input x_t and the previous hidden states h_{t-1}. Several gates control how the current hidden state

h_t changes: The forget gate f_t, the input gate i_t, and the output gate o_t. These gates are defined as follows [97]:

$$\begin{aligned} f_t &:= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\ i_t &:= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\ g_t &:= \psi(W_{xg}x_t + W_{ho}h_{t-1} + b_g) \\ o_t &:= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\ c_t &:= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &:= o_t \odot \psi(c_t) \end{aligned} \tag{8.3}$$

Where W_{xj}, W_{hj} are weight matrices and b_j is a bias vector for $j \in \{f, i, g, o\}$. The initial cell and hidden states are $c_0 = h_0 = 0$. The symbol σ refers to the logistic sigmoid function, ψ denotes the hyperbolic tangent, and $\odot$ is the element-wise multiplication. For a classification task, we add a discriminative layer after the last activation vector h_T to obtain class activation scores using a weight matrix W:

$$S_c^{\omega}(e) := W_c h_T \tag{8.4}$$

Since the ith hidden state vector h_i is updated by prior hidden states, previous sequence elements $\epsilon_1, \epsilon_2, ..., \epsilon_{i-1}$ are already taken into account in the evaluation of h_i. Thus, $S_c^{\omega}(e_i) = W_c h_i$ describes the accumulated class activation of the first $1 \leq i \leq E$ word embeddings $e_i := (\epsilon_1, \epsilon_2, ..., \epsilon_i)$ of an input $e = (\epsilon_1, ..., \epsilon_i, \epsilon_{i+1}, ..., \epsilon_E)$. To assess the contribution of a single word ϵ to the final class prediction, we decompose the final class activation score into the sum of the contributions of several individual words [97]:

$$S_c^{\omega}(e) := \sum_{i=1}^{E} S_c^{\omega}(e_i) - S_c^{\omega}(e_{i-1}) = \sum_{i=1}^{E} W_c(h_i - h_{i-1}) \tag{8.5}$$

The calculation of word contribution relies on evaluating the slope of the sequence of class activation scores, denoted as $S_c^{\omega}(e_1), S_c^{\omega}(e_2), ..., S_c^{\omega}(e_E)$. The slope is calculated as the difference between consecutive elements of the sequence. Since even the first word ϵ_1 within a text can significantly contribute to a certain class, we treat e_0 as a neutral placeholder that is not part of e and does not contribute disproportionately to any class; thus, $S_c^{\omega}(e_0) \approx \frac{1}{C}$. Technically, e_0 can be realized by pre-padding.

When modeling uncertainty with variational Bayes, the class activation score is computed as the average of T stochastic forward passes, that is $S_c(e) := T^{-1} \sum_{t=1}^{T} S_c^{\omega_t}(e)$. Computing the mean posterior probability $p(y = c|e_i, D)$ for each word sequence up to index i with $1 \leq i \leq E$ using MCD allows us to assess the development of uncertainties along the input sequence. Analogous to Eq. 8.5, we measure the word-level uncertainty U as the change in uncertainty contributed by a single word:

$$U(\epsilon_i) := U(e_i) - U(e_{i-1}) \tag{8.6}$$

Additionally, we derive the relevance R_c of each word in terms of its contribution to the class activation score. The relevance of a word is calculated as the class activation contributed by a single word compared to its previous sequence:

$$R_c(\epsilon_i) := S_c(e_i) - S_c(e_{i-1}) \tag{8.7}$$

According to Eqs. 8.5 and 8.7, the posterior probability of a word sequence e_i can also be calculated from the word relevance:

$$\begin{aligned} p(y = c|e_i, D) &:= \phi(S_c(e_i)) \\ &= \phi\Big(\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{i} S_c^{\omega_t}(e_j) - S_c^{\omega_t}(e_{j-1})\Big) \\ &= \phi\Big(\sum_{j=1}^{i} S_c(e_j) - S_c(e_{j-1})\Big) \\ &=: \phi\Big(\sum_{j=1}^{i} R_c(\epsilon_j)\Big) \end{aligned} \tag{8.8}$$

Since the direction of the certainty, e.g., whether a text contains a positive or negative sentiment at some index i, is much more meaningful than simply that the model is certain, we also suggest using the *Directed Uncertainty Explanation* (DUX) as a type of explanation. The DUX is based on the sequence uncertainty $U(e_i)$ and the sequence relevance $\phi(S_c(e_i))$ and is defined as:

$$\mathrm{DUX}(e_i) := \begin{cases} \phi(S_c(e_i)) & \text{if } U(e_i) \leq \tau_U \\ U(e_i) \end{cases} \tag{8.9}$$

The sequence relevance $\phi(S_c(e_i))$ is used if the model is certain about e_i. Otherwise, the sequence uncertainty $U(e_i)$ is taken. As the uncertainty threshold τ_U we use the mean of the uncertainty range, which is $\tau_U = 0.125$.

Overall, BayLUXT supports the following types of explanation: the word relevance $R_c(\epsilon_i)$, the sequence relevance $\phi(S_c(e_i))$, the word uncertainty $U(\epsilon_i)$, the sequence uncertainty $U(e_i)$, and the directed uncertainty explanation $DUX(e_i)$.

8.2.4 Visualizing Attributions

As previously outlined, a common approach to making users aware of word attributions is to rely on heat-maps. These encode the magnitude of word attributions as colors. To compute a word-level attribution mask for visualization via a heat-map, the attribution associated with the corresponding embedded words $\epsilon_1, \epsilon_2, ..., \epsilon_E$ must be traced back to their original natural language representation. Several steps must be performed to trace back the attribution scores.

So far, we have only considered the meaningful parts of the feature space given an input $\boldsymbol{e}$. However, sequences of word embeddings are generally padded to fit the same sequence length. Thus, the actual input has the form $(\rho_1, \rho_2, ...\rho_P)||(\epsilon_1, \epsilon_2, ..., \epsilon_E)$ where $\rho_1, \rho_2, ...\rho_P$ represent padded elements that hold no significant meaning for the classification outcome. Relevant sequence information only exists in the unpadded word embeddings $\epsilon_1, \epsilon_2, ..., \epsilon_E$. Therefore, attributions associated with padded elements need to be dropped. Second, cleaning steps applied to the input text, such as lowercase transformation or the removal of punctuation marks and stop words, must be traced back to preserve the original format of the input. Finally, for enhancing visual clarity, word attributions are typically normalized to sharpen the visualization, for instance, by using the absolute maximum attribution score per input sequence [22].

Heat-maps often become cluttered when visualizing large amounts of information. Therefore, we also explore the use of line graphs to represent the word attributions. This approach plots word indices on the x-axis, while attribution scores are plotted on the y-axis. Although the change in attribution scores is clearly visible in this representation, the direct relationship to individual words is less obvious. However, since the subsequent evaluation is primarily concerned with changes in attribution scores rather than the text itself, this does not diminish its effectiveness.

8.3 Study Design

We outline the research questions and evaluation criteria for the experimental and human evaluation. We describe the implementation details of BayLUXT and the datasets used, as well as the participants and procedure of the human evaluation.

8.3.1 Research Questions

We will investigate BayLUXT by examining the following research questions:

RQ1: Can BayLUXT provide a meaningful decomposition of the posterior probability of an input over the sequence of its words?

BayLUXT aims to offer profound insight into the decision-making of text classification models. We explore the ability of BayLUXT to decompose and extract sequence information, specifically how the posterior probability of the model evolves given various input texts.

RQ2: Is BayLUXT suitable for explaining the relevance and uncertainty of words in the context of text classification?

Traditional local explanation techniques offer word attributions that indicate the most relevant words for specific class outcomes. BayLUXT extends this by decomposing the prediction uncertainty at the word-level. We assess how well BayLUXT can distinguish between relevant, irrelevant, certain, and uncertain words or phrases with respect to the classification objective.

RQ3: Do explanations of uncertainty improve human understanding of text classification results?

It is currently unknown to what extent explanations of uncertainty provide insights for human users seeking to better understand classification models, compared to providing only traditional word relevance.

RQ4: How does explaining uncertainty compare to the traditional approach of explaining relevance?

We investigate how uncertainty explanations complement traditional relevance-based explanations. Our goal is to determine whether uncertainty explanations can replace or meaningfully enhance relevance explanations.

8.3.2 Evaluation Criteria

We evaluate the suitability of BayLUXT through two consecutive studies. First, we examine the effectiveness of BayLUXT through an experimental evaluation. Second, we conduct a human evaluation in which participants are presented with explanations of BayLUXT and are given a series of tasks, and a questionnaire.

Experimental Evaluation. The experimental evaluation assesses the correctness of BayLUXT. We investigate the explanatory power of BayLUXT on various text inputs. Since BayLUXT can explain text classification regardless of the actual domain and task, we focus on the common and widely studied task of sentiment analysis.

To answer RQ1 and RQ2, we first examine whether BayLUXT can detect sentiment changes within a text. For instance, a movie review might start with positive aspects and conclude with negative ones. A decomposition of the posterior probability $p(y = c|e, D)$ should be able to detect such sentiment changes. To systematically investigate the ability of BayLUXT to explicitly reveal changes in sentiment, we concatenate different texts with known labels. For instance, a text with a positive sentiment is concatenated with a text expressing a negative sentiment. We then examine whether the transitions between the different sentiment sections of the concatenated input are visible in the explanation. Second, we examine which words are frequently deemed uncertain or relevant for positive and negative sentiment texts and whether they are contextually meaningful. To accomplish this, we rank the words based on their mean attribution scores for positive and negative class outcomes and examine the sensibility of the most frequently occurring words. Third, we visualize the attribution scores provided by BayLUXT. Given several input examples, we investigate which words are highlighted by different types of explanations and demonstrate the insights provided.

Human Evaluation. To answer RQ3 and RQ4, we conducted a human evaluation of BayLUXT. Since it is overwhelming to provide five different types of explanations to human users to facilitate their understanding of a prediction, we focus on the evaluation of DUX. DUX is considered because we believe it provides the most comprehensive insight into assessing the uncertainty of a prediction. We asked participants to complete several comprehension tasks of text classification explanations provided by BayLUXT, covering both traditional relevance-based explanations and DUX. Participants were also asked to complete a short questionnaire.

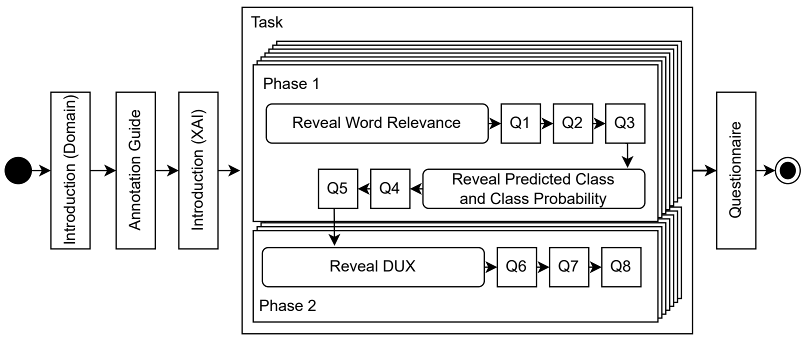


Figure 8.1: Overview of the human evaluation design.

The task of the human evaluation, as shown in Figure 8.1, is divided into two parts. In total, both tasks are performed on 7 text instances. Given a text instance, BayLUXT is used to classify it and provide a traditional word-relevance-based explanation $\phi(S_c(\epsilon_i))$, as well as DUX(e_i). In each task, participants are asked several questions to investigate the usefulness of the two types of explanations.

- **Part 1: Explaining Relevant Words**. In the first part of the user evaluation, participants are shown a text and an explanation that highlights the most relevant words for positive and negative sentiment. Based on the explanations alone, participants are first asked to decide whether the text has a positive or negative sentiment (Q1). If no clear decision can be made, an instance can also be marked as undecidable. Second, without knowing the predicted class label, participants were asked how much the provided explanation made sense to them (Q2). Participants were then asked how easy it was to make a clear classification decision (Q3). In the next step, participants were shown the classification outcome y and the class probability $p(y|e)$. They were asked how trustworthy they thought the predictions were (Q4). Finally, we asked participants to what extent the explanation for that class made sense, i.e., supported the classification result (Q5). Our study does not reveal the ground truth label, as this information would not be available in a real-world scenario.

- **Part 2: Explaining Uncertainties via DUX**. The second part of our human evaluation concerns the suitability and validity of DUX. Participants are presented with DUX. Given the same text, we ask how understandable the provided explanation of the underlying uncertainties was (Q6). Participants were then asked to what extent the explanation of the uncertainty helped them to judge the trustworthiness of the prediction (Q7). Finally, participants were asked which type of explanation was most helpful in understanding the prediction outcome and its confidence (Q8). Participants were given a choice between "word relevance", "uncertainties", "both equally" or "neither". For Q2 to Q7, participants could express their agreement or disagreement with the question using a 5-point Likert scale.

Finally, participants are asked to answer a series of questions covering various aspects of explainability in general and a comparison of the two types of explanations.

8.3.3 Datasets

To demonstrate the capabilities and correctness of BayLUXT, we apply it to the well-studied task of sentiment analysis [243]. Sentiment analysis has the advantage that explanations are easy to understand and do not require specific expert knowledge. As a technical contribution to the field of explainable text classification, no serious differences in the basic functioning of BayLUXT are to be expected with respect to other datasets and domains. Only the complexity of the task impacts the classification performance of a model and thus the explanation associated with it. For our evaluation, we consider the IMDB dataset [243], which consists of 50,000 polarized movie reviews, equally divided into positive and negative reviews according to their sentiment. In our experiments, we use 25,000 examples for training, 12,500 for validation, and 12,500 for testing. After training, we achieve a test accuracy of 88%.

We also rely on the same sentiment analysis task and trained LSTM model for the human evaluation. We consider a total of seven text instances for the human evaluation. We randomly sampled four highly uncertain reviews from the test dataset. Additionally, we randomly sampled one positive and one negative text with very low uncertainty. In addition, we used a text with a mixed sentiment written by ourselves. Since the movie reviews are usually very long and time-consuming to read, we simplified and reduced the sampled texts to include less than 60 words.

8.3.4 Implementation Details

We implemented BayLUXT using an LSTM with an additional dropout-layer after the embedding-layer with a dropout probability of 0.5. We consider the LSTM configuration used in the official Tensorflow example[2]. We take pre-trained Google word2vec embeddings[3] with a dimensionality of 300 as our word representations. Further, we use the Adam [196] optimizer, a batch size of 50 and apply early stopping. For each input, we consider $T = 50$ model runs to approximate the posterior probabilities.

8.3.5 Participants

We recruited participants from students participating in an advanced ML project. All participants were familiar with back-box classifiers for text classification and the interpretation of traditional relevance-based explanations. Furthermore, all

[2]https://keras.io/examples/imdb_lstm/
[3]https://code.google.com/archive/p/word2vec/

participants had at least two years of experience in software development. A total of 15 participants were recruited for the human evaluation. One participant was excluded from the results due to a high number of incomplete responses during the task.

8.3.6 Procedure of the Human Evaluation

The participants were invited to carry out the experiment in person. All tasks were distributed on paper. First, the structure of the human evaluation was explained, as shown in Figure 8.1. Then the overall problem domain of explanations as a means to achieve more transparent predictions was briefly introduced. An annotation guide for sentiment analysis was provided, outlining the overall objective and how to distinguish between the two classes. We presented an example of a clearly negative and a clearly positive review. We also provided examples of terms such as "*perfect*", "*great*" and "*awful*" as typical indicators of a positive or negative review. Next, we explained the process of comprehending relevance-based explanations and DUX by showing an example of each type of explanation. We also provided a step-by-step guide for understanding word relevance and DUX. Finally, each participant was given a task sheet and the questionnaire.

8.4 Experimental Evaluation Results

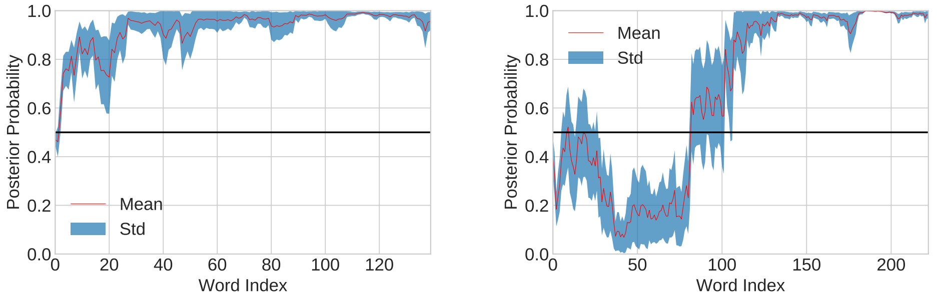
This section presents the experimental evaluation of the technical contribution of BayLUXT. This includes its ability to decompose the posterior probability and uncertainties over the input sequence and the attribution of individual words. Our implementation and experimental results are publicly available online[4].

8.4.1 Sequence Decomposition

First, we investigate the information provided by the decomposition of posterior probabilities and prediction uncertainties. In particular, given some input texts, we experimentally assess the degree of transparency that BayLUXT reveals.

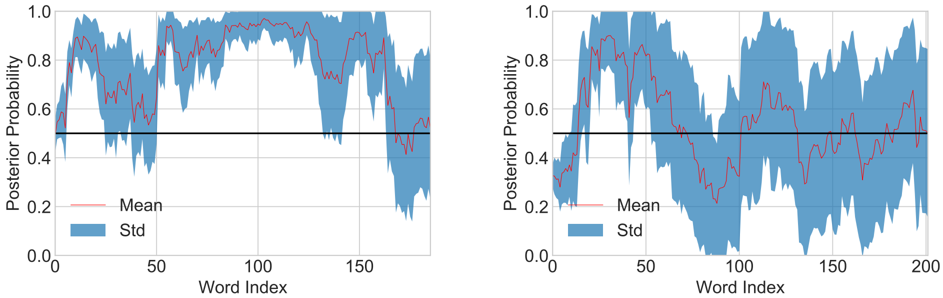
Figure 8.2 gives an example of two movie reviews that are clearly classified as negative. Both posterior probabilities are close to one. The x-axis refers to the word index i of the processed subsequence e_i. The y-axis shows the posterior probability for the negative sentiment case. The graph can be interpreted as the prediction outcome if the model only considers the first i word embeddings of the input.

[4] `https://github.com/jsandersen/WU-RNN`



(a) (b)

Figure 8.2: Two predictions with similar class outcomes, but different sequential behavior. The path of the mean posterior probability reveals relevant regions in the input.



(a) (b)

Figure 8.3: Two highly uncertain predictions with similar class probabilities but different sequential behavior. The path of the mean posterior probability reveals relevant regions in the input.

Decomposing the posterior probabilities reveals different model behavior along the input sequences. In Figure 8.2a the model has inferred early that the input describes a negative review. In Figure 8.2b, the input is first evaluated as positive and then changes to a negative sentiment classification result. Two examples of highly uncertain predictions are depicted in Figure 8.3. The example in Figure 8.3a is predominantly perceived as a negative sentiment. Towards the end, however, the model loses confidence in its decision. The correct classification (ground truth) for this example would be a positive sentiment. In contrast, for the example shown in Figure 8.3b, the model is not sure which class to prefer for almost every index. The model fluctuates between the two classes and finally makes a decision with a posterior probability of 0.51.

The path of the mean posterior reveals the structure of the input. BayLUXT demonstrates its ability to find indistinguishable positive or negative sections

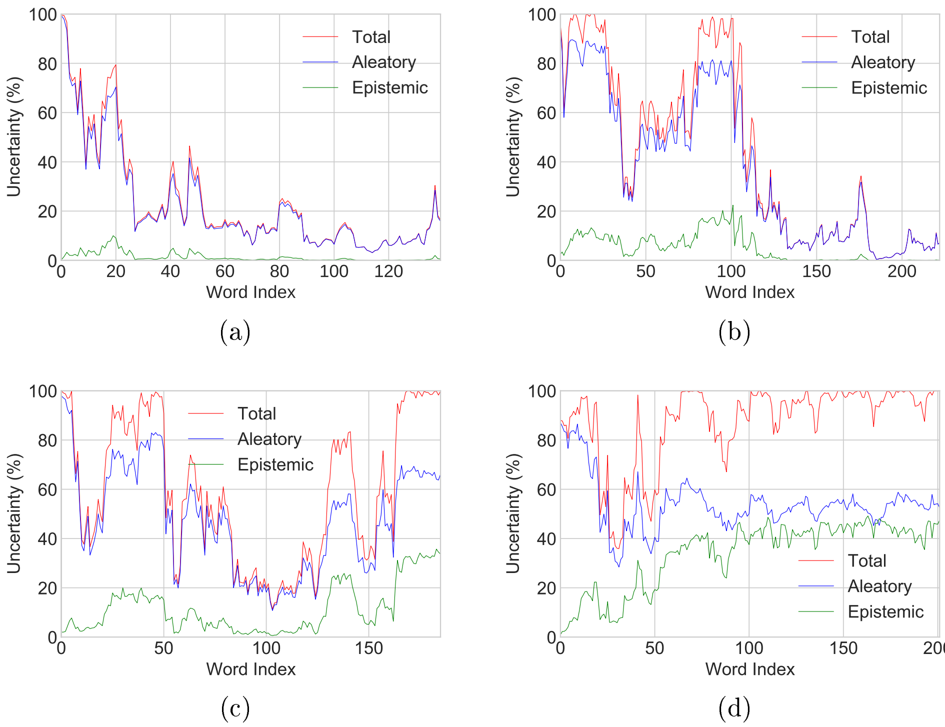
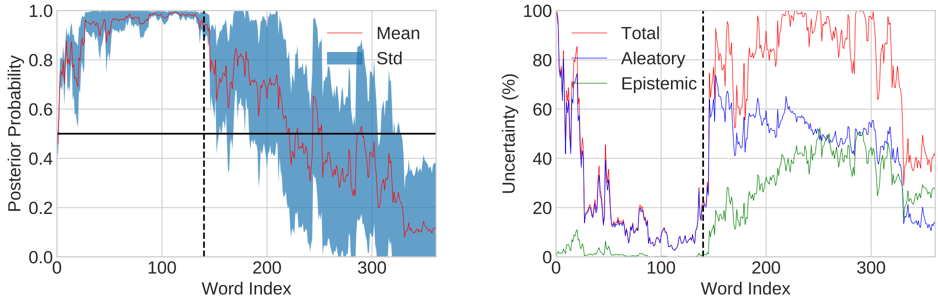


Figure 8.4: Decomposition of the total, aleatory, and epistemic uncertainties for the examples used in Figure 8.2 (a and b) and Figure 8.3 (c and d).

within the input text. With traditional uncertainty estimation approaches, these internal insights would remain hidden because they provide a single uncertainty value that covers the entire input text.

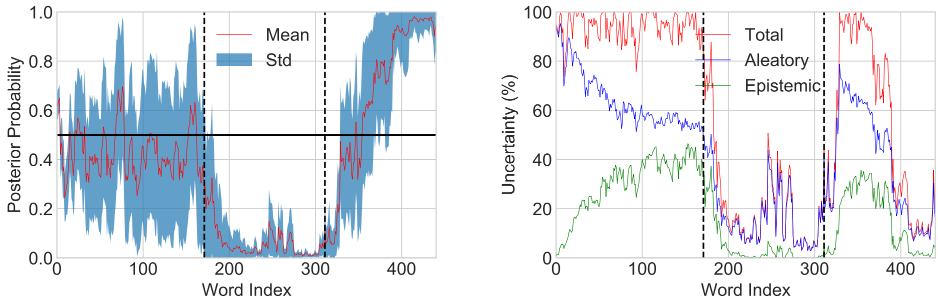
Figure 8.4 plots the total uncertainty as well as its aleatory and epistemic components for the posterior distributions shown in Figures 8.2 and 8.3. The total uncertainty is limited to a maximum of 0.25 (100%). In the example shown in Figure 8.4a, the model starts to have a high uncertainty when only a few words are considered. In comparison, in Figure 8.4b, the model is much more uncertain in the first half of the text. In both cases, however, the uncertainty is almost eliminated by considering additional sequence information. In contrast, the examples in Figures 8.4c and 8.4d are almost maximally uncertain.

To provide a more detailed example, we concatenate a clearly negative review of 140 words with a clearly positive review of 239 words. As shown in Figure 8.5a, the first 140 words are clearly identified as negative with a total uncertainty of 24%. At the beginning of the second review, the mean posterior probability drops and becomes highly uncertain. Furthermore, Figure 8.5 shows that the uncertainty increases as the sentiment shifts. Another example is shown in Figure 8.6, where an uncertain, a positive, and a negative review are concatenated.



(a) (b)

Figure 8.5: A negative review of 140 words concatenated with a positive review. The figure shows the effect of a changing sentiment along the input.



(a) (b)

Figure 8.6: Concatenation of a highly uncertain, positive, and negative review with 140, 171, and 140 words, respectively.

Again, the decomposed posterior probability shows the change in sentiment that matches that of the concatenated review. Overall, our experiments show that the decomposition of NN outputs can provide valuable information to support the understanding of classification decisions.

8.4.2 Relevant and Uncertain Words

Next, we examine the most relevant words for each class outcome to demonstrate the correct decomposition of the class activation score S_c given some input x. It is assumed that words with positive meanings such as "*brilliant*", "*favorite*", or "*excellent*", are often used to describe positive movie reviews. Conversely, negative words such as "*boring*" or "*awful*" indicate negative reviews. Table 8.1 lists the most relevant words on average for the positive and negative sentiment class outcomes on the test dataset. Given a word embedding ϵ, its average relevance $\bar{R}_c(\epsilon)$ is calculated as the ratio of the sum of word activations to its total frequency. The inferred words match our expectations, indicating that our

	Positive	$\bar{R_c}$	Count	Negative	$\bar{R_c}$	Count
1	perfect	1.21	720	worst	1.55	1237
2	excellent	1.14	974	disappointing	1.35	179
3	superb	1.05	303	unwatchable	1.31	57
4	refreshing	1.04	79	insult	1.22	120
5	finest	0.99	135	waste	1.22	646
6	favorites	0.97	92	forgettable	1.21	84
7	favorite	0.95	537	awful	1.18	833
8	delightful	0.94	91	plagiarized	1.15	1
9	interweaving	0.91	1	disappointment	1.15	178
10	unforgettable	0.90	68	appalling	1.12	72
11	excellently	0.90	17	pointless	1.12	230
12	unsurpassed	0.90	5	uninspired	1.11	82
13	celebrates	0.90	9	stinker	1.10	59
14	timeless	0.90	51	torpid	1.10	1
15	enjoyable	0.89	383	twit	1.10	2

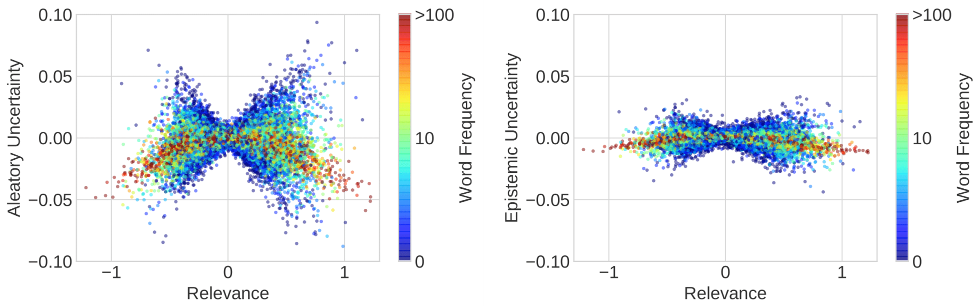
Table 8.1: Top 15 most relevant words for positive and negative reviews.

	Aleatory	$\bar{U_a}$	Count	Epistemic	$\bar{U_e}$	Count
1	**worst**	-0.079	1237	<u>dumber</u>	-0.019	21
2	**awful**	-0.058	833	ineffective	-0.017	17
3	<u>**excellently**</u>	-0.056	17	instrumental	-0.016	11
4	<u>dumber</u>	-0.055	21	vapid	-0.015	16
5	<u>stink</u>	-0.053	12	crow	-0.015	16
6	**waste**	-0.052	646	<u>**excellently**</u>	-0.014	17
7	<u>puerile</u>	-0.049	13	<u>stink</u>	-0.014	12
8	**insult**	-0.049	120	lifeless	-0.013	35
9	**excellent**	-0.048	974	mediocrity	-0.013	28
10	**finest**	-0.047	135	gutter	-0.013	11
11	**disappointing**	-0.047	179	<u>mst3k</u>	-0.013	81
12	mesmerizing	-0.047	24	clowns	-0.013	11
13	**stinker**	-0.047	59	<u>puerile</u>	-0.013	13
14	apologize	-0.047	11	wasting	-0.013	56
15	<u>mst3k</u>	-0.047	81	hatchet	-0.013	21

Table 8.2: Top 15 words that reduce the overall aleatory and epistemic uncertainty of a prediction. Words in bold are among the 15 most relevant words of both classes (Figure 8.1). Words that strongly reduce both epistemic and aleatory uncertainty are underlined.

model can correctly infer the meaning of emotional phrases. No unintuitive or ambiguous words were noticed.

Table 8.2 shows which words of the test data, on average, reduce the aleatory or epistemic uncertainty of a prediction the most. It is noticeable that the majority of the listed words that reduce aleatory uncertainty are also the most sensitive (as listed in Table 8.1). The sequence decomposition has demonstrated that a prediction begins to have almost maximum aleatory uncertainty when only the first word of the text is considered. At this point, the epistemic uncertainty is close to zero. Furthermore, by definition, relevant words increase the class activation score the most. Therefore, it is reasonable that relevant words will reduce aleatory uncertainty the most, especially at the first encounter.



(a) (b)

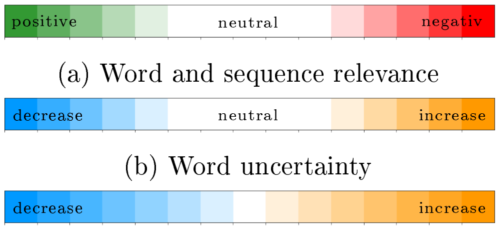
Figure 8.7: Dependencies between word relevance and a) aleatory and b) epistemic uncertainty.

Figure 8.3 displays the relationship between the average word relevance and the uncertainty contributed by each word. For each word ϵ the y-axis denotes the word relevance $R_c(\epsilon)$ and the x-axis refers to the aleatory and epistemic uncertainty. The plot reveals that relevant words are more likely to add or remove uncertainty in the model. This was already seen in Table 8.2. Furthermore, uncommon words are likely to add the most uncertainty, while the most common words reduce the uncertainty. A comparison of the figures reveals similar behavior for aleatory and epistemic uncertainty. However, the overall model is less affected by epistemic uncertainty than by aleatory uncertainty.

8.4.3 Text Explanations

To visualize the degree of word relevance and uncertainty, we use heat-maps and a gradient-based color scheme. The color gradients used for sentiment analysis are shown in Figure 8.8. Positive sentiment words are highlighted in green and negative sentiment words are highlighted in red. In addition, words that reduce uncertainty are highlighted in blue and words that increase uncertainty are highlighted in orange. Neutrality is illustrated by keeping the background color white (no highlighting). As neutrality can be interpreted to some extent as ambiguity and uncertainty, we use a modified gradient for sequence uncertainty that does not include a broad neutral state.

Table 8.3 illustrates all types of explanations provided by BayLUXT. More text examples are shown in Table B.1. These are the word relevance $R_c(\epsilon_i)$, the sequence relevance $\phi(S_c(e_i))$, the word uncertainty $U(\epsilon_i)$, the sequence uncer-



(a) Word and sequence relevance

(b) Word uncertainty

(c) Sequence uncertainty

Figure 8.8: Meaning of the color gradients of BayLUXT for different types of explanations.

Label and Confidence	Text and Explanations	Type
Positive 0.66	It was an excellent performance by the actors and a great setting. Unfortunately, the plot was terrible. I hope that the actors find new projects.	$R_c(\epsilon_i)$
	It was an excellent performance by the actors and a great setting. Unfortunately, the plot was terrible. I hope that the actors find new projects.	$\phi(S_c(e_i))$
	It was an excellent performance by the actors and a great setting. Unfortunately, the plot was terrible. I hope that the actors find new projects.	$U(\epsilon_i)$
	It was an excellent performance by the actors and a great setting. Unfortunately, the plot was terrible. I hope that the actors find new projects.	$U(e_i)$
	It was an excellent performance by the actors and a great setting. Unfortunately, the plot was terrible. I hope that the actors find new projects.	$DUX(e_i)$

Table 8.3: Visualizations of the different types of explanations obtained from BayLUXT. A word at index i is highlighted according to the word relevance $R_c(\epsilon_i)$, the sequence relevance $\phi(S_c(e_i))$, the word uncertainty $U(\epsilon_i)$, the sequence uncertainty $U(e_i)$, and DUX $DUX(e_i)$. Positive and negative sentiment words are highlighted in green and red, respectively, while words that reduce and increase uncertainty are highlighted in blue and orange.

tainty $U(e_i)$ and $DUX(e_i)$. The example shown is classified as positive with a mean posterior probability of 0.66. The explanations given can be interpreted as follows:

- **Word Relevance** – $R_c(\epsilon_i)$**.** The words "*excellent*" and "*great*" are identified as the most relevant words for a positive sentiment, while "*unfortunately*" and "*terrible*" are negative. Considering only these words, it may not be obvious why the classifier reports positive instead of negative as the class result.

- **Sequence Relevance** – $\phi(S_c(e_i))$**.** This explanation highlights a word in terms of the posterior probability that the classifier would report if only the text from the beginning to that word were considered by the classifier. Thus, the ith word shows the classification outcome if only the first i words were passed to the classifier. For example, the word "*excellent*" is highlighted in green as the text "*I was an excellent*" was recognized as positive. It can be seen that the classifier would provide a positive sentiment outcome with high confidence until the negative word "*terrible*" appears.

- **Word Uncertainty** – $U(\epsilon_i)$**.** The word uncertainty shows the total uncertainty contributed by each word. It can be seen that the word "*excellent*" reduces the classification uncertainty, as it is a highly relevant word that leads to high sequence relevance. When the contradictory relevant word "*unfortunately*" appears in the sequence, the class probability decreases, resulting in a loss of confidence and an increase in uncertainty.

- **Sequence Uncertainty** – $U(e_i)$**.** The sequence uncertainty, analogous to the sequence relevance, illustrates the uncertainty of the prediction when only the first i words of the input have been considered by the model. The model is initially highly uncertain once it observes a relevant term. If a relevant term, i.e., "*unfortunately*", occurs that contradicts the current sequence relevance, the sequence uncertainty increases. Finally, the model remains highly uncertain about the overall prediction in this example.

- **DUX** – $DUX(e_i)$**.** In the case of uncertainty, the DUX is equivalent to the sequence uncertainty. Certain words are highlighted in the color of the sentiment they are relevant to.

In this example, the model remains highly uncertain about the overall prediction. The following observations can be made from the examples:

1. Text classifiers have a high degree of uncertainty in making a clear prediction at the beginning of a text when no relevant term has yet appeared.

2. Highly relevant terms (positive or negative) reduce the uncertainty if the classifier is highly uncertain beforehand.

3. If two highly relevant terms for the same class occur consecutively in a text, the second term is likely to further reduce the uncertainty of the prediction.

4. If two highly relevant but contradictory terms occur consecutively in a text, the second term increases the uncertainty of the prediction.

5. The same word can add or remove uncertainty to a prediction depending on its position in the text.

8.5 Human Evaluation Results

The following section outlines the results of the human evaluation of BayLUXT. The texts, classification results, and explanations provided to the participants are shown in Table B.2.

8.5.1 Explaining Relevant Words

In the first phase of the user evaluation, participants are shown a text and a word relevance explanation that highlights the most relevant words associated with positive and negative sentiments. Initially, participants are requested to determine whether the sentiment of the text is positive or negative based solely on the explanation provided. If no clear decision can be made, participants can label an instance as unknown. The label assignments provided by the participants for each of the seven texts are illustrated in Figure 8.9.

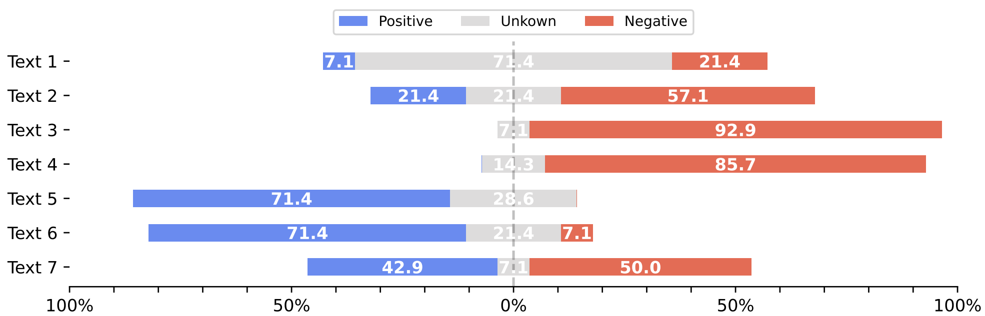
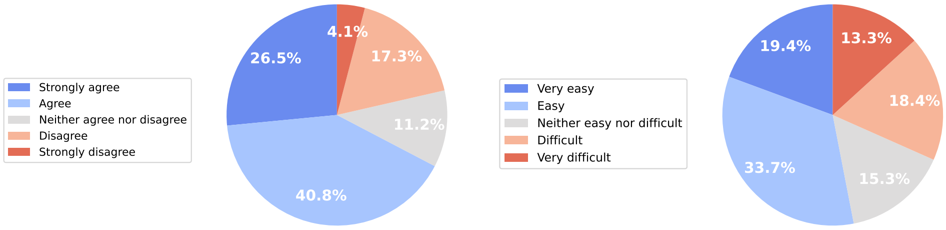


Figure 8.9: Responses to Q1: “Based on the word relevance, determine whether the following text expresses a more positive or negative sentiment”.

The results indicate that assigning labels to the text instances posed a considerable challenge. In the majority of cases (4 out of 7), conflicting labels were assigned. In addition, on average 24.5% of the participants could not clearly decide which class a text belonged to. Text 7 notably stands out, where 92.9% of the participants could clearly decide on its class. However, only 50% of the participants agreed on the most common class.

Second, participants were asked to rate the coherence of the highlighted words without prior knowledge of the predicted class label. A significant majority of 67.3% either agreed or strongly agreed that the explanations provided made sense (Figure 8.10a). In contrast, only 21.4% disagreed or strongly disagreed.



(a) Q2: "Did the words maked as relevant make sense to you?".

(b) Q3: "How easy was it for you to reach a clear decision based on the marked words?".

Figure 8.10: Responses to Q2 and Q3.

Regarding the ease of making a clear decision, 53.1% stated that it was easy or very easy (Figure 8.10b). However, 31.6% found it difficult or very difficult to choose between the three possible alternatives (positive, negative, undecided). Thus, on average, participants found it more difficult to make a clear decision based on an explanation than to understand the explanation itself.

In the next phase, participants were presented with the classification results and the associated confidence levels. When asked about the perceived trustworthiness of the predictions, on average 53.1% indicated high or moderate trustworthiness, while 24.5% indicated a low or no trustworthiness (Figure 8.11). Texts 1 and 6 were met with considerable skepticism, with only 78.5% and 71.4%, respectively, finding these predictions untrustworthy. None of the predictions received absolute confidence from all participants. The predictions of texts 4 and 5 proved to be the most trustworthy, each receiving agreement from 71.4% of the participants. Notably, these predictions also had the highest confidence levels at 97% and 91%, respectively.

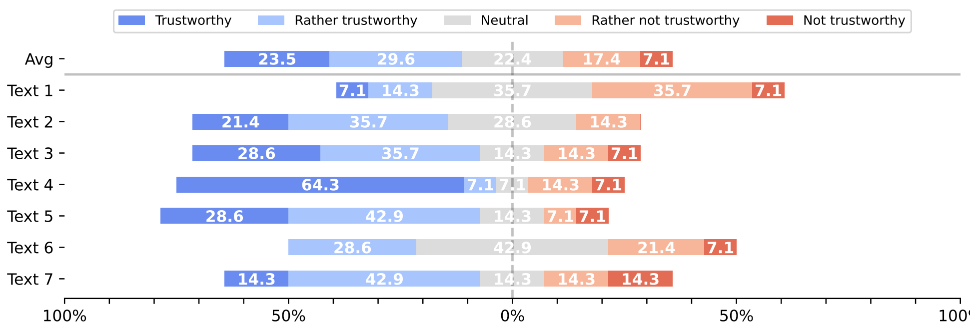
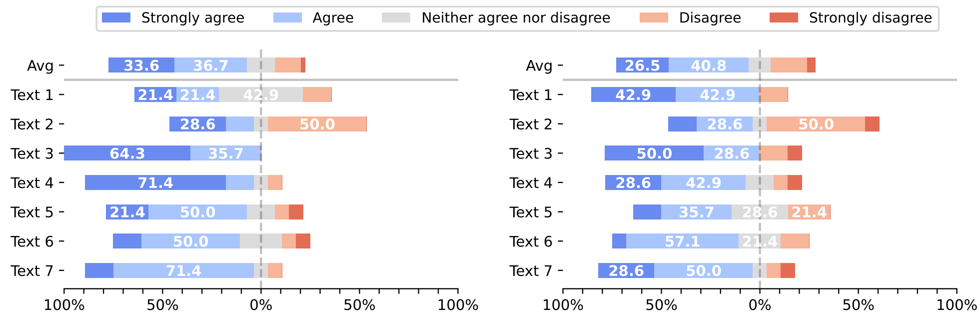


Figure 8.11: Responses to Q4: "How trustworthy do you consider this prediction to be?".



(a) Coherence of the word relevance without knowing the class label (Q2).

(b) Coherence of the word relevance with knowing the class label (Q5).

Figure 8.12: A comparison of the coherence between relevance-based explanations without (Q2) and with (Q5) knowing of the class label.

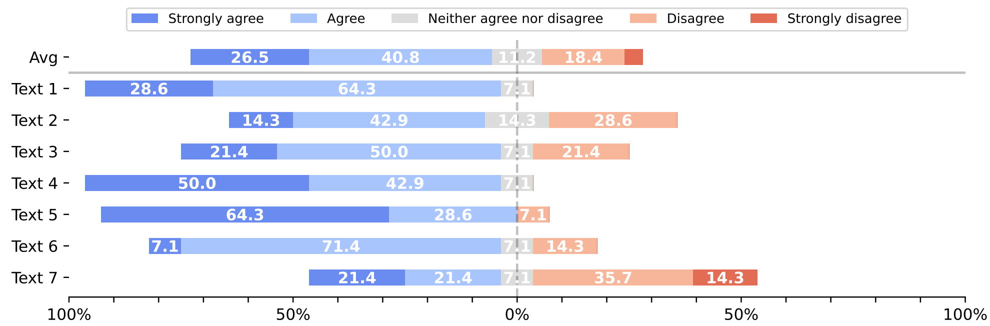


Figure 8.13: Responses to the Q6: "The markings of uncertainty appear understandable to me".

Participants were then asked to rate the extent to which the explanation for that class made sense and supported the classification result (Figure 8.12a). The results indicate that, on average, the coherence of the explanation improves when participants know the class outcome and its confidence. The comparison of coherence with and without knowledge of the predicted class and its confidence level is illustrated in Figure 8.12b. The most significant increase in coherence was observed for Texts 4, 3, and 5, with improvements of 47.6%, 47.4%, and 12.5%, respectively. Conversely, the coherence of Text 1 decreased by 25.7%.

8.5.2 Explaining Uncertainties

The second segment of our human evaluation concerns the adequacy of uncertainty explanations. Figure 8.13 shows the responses regarding how understandable the provided uncertainty explanations were. On average, 75.5% of the participants stated that the uncertainty explanations were at least somewhat understandable, while on average 17.3% expressed that they were not

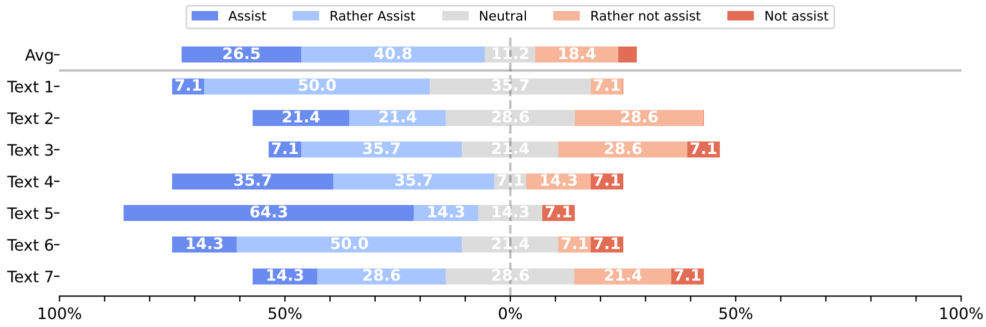


Figure 8.14: Responses to Q7: "To what extent does the uncertainty assist you in assessing the trustworthiness of the prediction?".

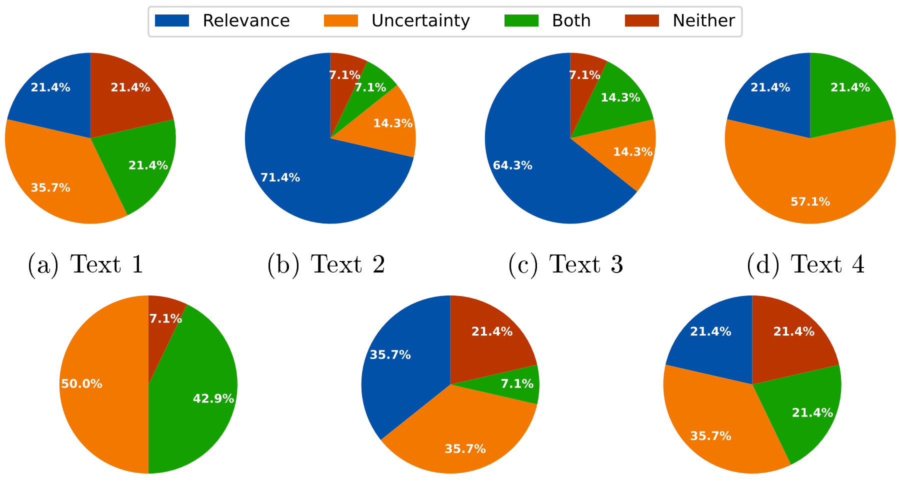
very understandable or not understandable at all. Thus, the comprehensibility of uncertainty explanations is only slightly lower than that of word relevance. However, the level of comprehensibility varied depending on the text being explained. Uncertainties in Texts 7, 2, and 3 were least understood, while those in Texts 1 and 4 were best understood.

In the next step, participants were asked to what extent the explanation of uncertainty contributed to their assessment of the reliability of the prediction. Figure 8.14 shows the responses. On average, 57.1% of participants indicated that the explanation of uncertainty helped them understand the reliability of the prediction, while about 20.4% stated that it did not help them.

Finally, participants were asked which type of explanation was most helpful in understanding the prediction and its confidence. Participants were given the options "relevance", "uncertainty", "both", or "neither". The answers are illustrated in Figure 8.15. The results show that there is no ideal explanation that consistently leads to a better understanding of a prediction. However, on average, 90.8% of participants found at least one of the two explanations helpful. In three cases (Text 1, 2, and 3), the word relevance was considered to be more helpful. In contrast, uncertainty was found to be more helpful in three other cases (Text 4, 5, and 7). Only for Text 6, both types of explanations were found to be equally helpful (35.7% of the votes each).

8.5.3 Questionnaire

Figure 8.16 shows the responses to the final questionnaire. When asked if the participants would blindly trust the model without any explanations, 57.1% disagreed, and 21.4% each were undecided or would rather blindly trust the model. However, 100% of the participants agree that the explanations generally help to understand and comprehend the artificial decisions. The statement



(a) Text 1 (b) Text 2 (c) Text 3 (d) Text 4

(e) Text 5 (f) Text 6 (g) Text 7

Figure 8.15: Responses to the Q8: "Which type of explanation was more helpful for you to understand the classification decision?".

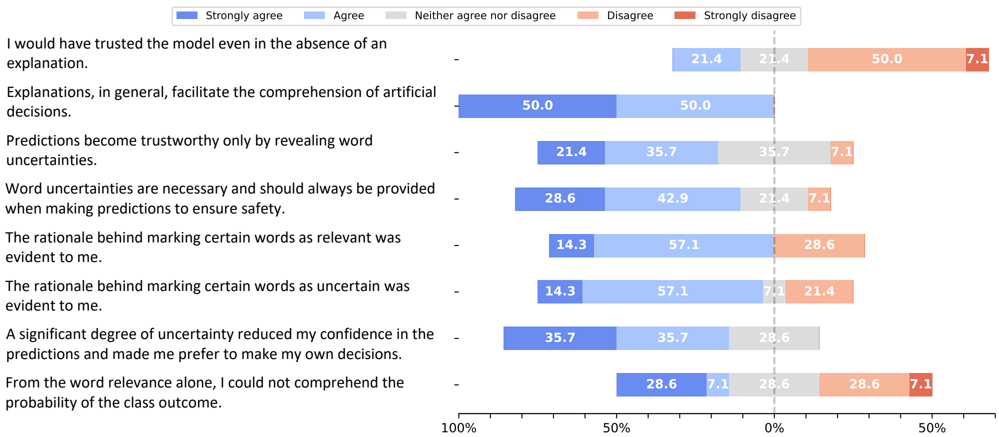
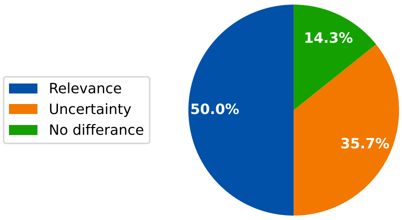
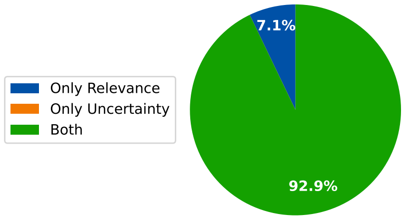


Figure 8.16: Responses to the questionnaire.

that uncertainties are necessary to make a prediction reliable was agreed to by 57.1%, while 7.1% would rather disagree with this statement. In addition, 71.4% believe that uncertainties should always be stated for the sake of certainty. Again, 7.1% disagree with this statement. Regarding the comprehensibility of the explanations, 71.4% said they understood why words were marked as relevant or uncertain. 28.6% were unclear about the relevance of the words, and 21.4% were rather unclear about the uncertainty. On the statement that it is better to decide manually when there is little confidence in a prediction, 71.4% agreed, and 28.6% were undecided. In response to the statement that using only word relationships gives a poor understanding of class probability, 36.7% agreed and 36.7% disagreed.



(a) "I found the following type of explanation more helpful ..."



(b) "The optimal type of explanations for me would be ..."

Figure 8.17: Comparison of relevance and uncertainty explanations regarding their usefulness for humans.

When questioned about the most useful types of explanations (Figure 8.17a), half of the respondents favored relevance explanations, while 35.7% found uncertainty explanations to be the most effective. A negligible 14.3% expressed no preference. The optimal type for explaining a prediction was considered by 92.9% to be the simultaneous explanation of relevance and uncertainty (Figure 8.17b). However, a minority (7.1%) considered only relevance-based explanations to be ideal.

8.6 Discussion

We discuss the implications of our findings, consider their applicability to various fields, discuss the limitations of our study, and review related work.

8.6.1 Implications

The research field of XAI for text classification has mainly focused on relevance-based explanation techniques, attempting to accurately extract which words contribute most to certain class outcomes. While these types of explanations have demonstrated valuable insights into black-box classifiers, they do not explicitly communicate prediction uncertainties. They lack the ability to make humans aware of uncertainties, a core principle of securing ML models. With BayLUXT, we propose one of the first frameworks to explain prediction uncertainties accordingly. We demonstrate that BayLUXT is sufficient to extract several types of explanations: word relevance, sequence relevance, word uncertainty, sequence uncertainty, and DUX. In comparison, previous local explanation techniques are mainly limited to word relevance.

Uncertainty awareness is very important when dealing with probabilistic and imperfect classifiers. It reveals weaknesses, misunderstandings, and ambiguities in the model or in the inputs that greatly affect the applicability of the classifier. Understanding the uncertainty of a model is particularly important for HiL systems, since humans typically have to deal with situations where the model is highly uncertain. Therefore, cognitive support to explain why uncertainties occur is desirable. Our experiments show that understanding a model's uncertainty helps human users to better comprehend a model's prediction, and in some cases is even more useful than traditional relevance-based explanations.

Our results also imply that uncertainty explanations harmonize with word-relevance-based explanations to aid the understanding of predictions. Our study indicates that 92.9% of participants would prefer to have access to uncertainty-based explanations in addition to word relevance to improve their understanding of text classifiers. Thus, uncertainty information does not replace word relevance, but complements it in terms of informativeness. Which type of explanation is considerably more useful depends on the input text and the model.

Although our investigation is limited in scope, covering only a small number of tasks with a total of 15 participants, our work provides initial promising findings that highlight the potential of uncertainty-based explanations in text classification. We demonstrate a clear benefit of uncertainty-based explanations for improving comprehension of text classification results. Trust in predictions increases with the specification of word uncertainties, indicating the need to include such uncertainties for safety and reliability. We argue that humans should be made aware of all available uncertainty information in high-risk tasks. Uncertainty awareness improves model comprehension and is likely to enhance the perception of model misbehavior.

Although our implementation of BayLUXT has limitations and our investigation is limited in the scope, we see BayLUXT as a first important step towards anchoring the concept of uncertainty explanations as an integral part of the XAI literature. So far, BayLUXT can be applied to any single-layer RNN-based text classifier without changing the architecture of the model. Since the explanations are extracted from the internal states of the recurrent layer, no additional computational effort is required during training. Only a Bayesian approximation must be performed to model uncertainties.

8.6.2 Field of Application

Explanations are a key enabler of the HiL approach. The use-case of BayLUXT is to make human users aware of uncertainties in their predictions that traditional XAI approaches could not reveal.

The primary goal of BayLUXT is to provide deeper insight into ML-based text classification. Uncertainty awareness is promising to improve the awareness of what a model can and cannot accomplish and help mitigate the risk of unconditional trust in unreliable results. A typical example is the Active Moderation and Decision Support System patterns. Here, human labelers typically deal with highly uncertain classification results that are likely to be wrong. Making these uncertainties explicit through BayLUXT provides cognitive support for understanding difficulties and potential misconceptions of the model, thus facilitating the labeling process.

Another application of BayLUXT is the debugging of ML-based text classifiers. Debugging in the context of ML [123] describes the process of controlling, identifying, and resolving problems or errors in an ML model's architecture, training data, trained parameters, or classification performance. Debugging involves rigorously testing the model on unseen data, systematically analyzing the model's behavior, examining intermediate results, and diagnosing problems in order to improve the model's classification performance. BayLUXT helps to understand how individual words contribute to prediction uncertainty. This improves insight into the model's reasoning process, including the source of uncertainty, potential misconceptions, knowledge gaps, and biases. Also of great interest are cases that are misclassified but actually have no uncertainty.

In addition, BayLUXT can be used to assess and improve the quality of text instances using prediction confidence as an indicator of data quality. Consider an issue tracker where users submit questions, bug reports, and feature requests. If a user wants to submit a bug report but chooses to formulate it as a question, this is considered a low-quality bug report. Similarly, a feature request is of poor quality if it describes bugs or is in the form of a question. In both cases, the text is of low accuracy and consistency. A trained ML model approximates the knowledge required to perform a particular task, i.e. text classification. Thus, given a class label, a classifier can judge how well a given text fits into that class. Deviations and inconsistencies explained by BayLUXT make the model's knowledge explicit and are likely to support users write better quality text.

8.6.3 Threats to Validity

We further outline the internal and external threats to validity.

Internal validity. First, we demonstrate the functional sufficiency and correctness of BayLUXT using a limited number of randomly sampled text instances from a hold-out dataset. Other samples may yield different results. Furthermore, since it is not feasible to prove the functional correctness of a local explanation, our investigation is qualitative, introducing a degree of subjectivity.

Although we relied on random sampling techniques to minimize sampling bias, we trimmed some of the text instances to be simpler and not overwhelmingly large within our human evaluation. This may make the samples less representative of the overall dataset. In addition, BayLUXT explanations are non-deterministic because they rely on Bayesian approximations. The number of forward passes affects the consistency and reliability of the results. To mitigate this problem, we perform 50 forward passes.

As with any human evaluation, our study may be corrupted by human biases that affect the interpretation and perception of explanations. Participants' prior knowledge, experience, or individual preferences may also influence their evaluation of explanations. In addition, all of our participants can be considered ML-experts, which might alter their behavior or responses due to their awareness of the technical aspects of ML systems. Domain experts without technical knowledge may have a different understanding of local explanations, leading to a bias in their human judgment.

External validity. The results of our human evaluation may have limited generalizability to other user populations or contexts. The number of participants in our user evaluation is limited, and the sample may not be fully representative of the full range of users across different domains and user groups, e.g., ML experts and domain experts.

Second, BayLUXT is only evaluated in the context of sentiment analysis, which limits the generalizability of the results. In sentiment analysis, class boundaries are likely to be fuzzy due to the subjectivity of sentiment characteristics. It may be that the results would be different for better-defined and less subjective tasks. However, we argue that it is more relevant to evaluate human judgments for more difficult tasks that could not easily be performed autonomously by an ML model (Chapters 5 and 6). In addition, we only applied BayLUXT to binary classification tasks. More classes could add complexity to the explanations and lead to different results.

Another limitation of our implementation of BayLUXT is its model-specific nature. BayLUXT is only applicable to RNN-based classifiers. Further research is needed to extract uncertainty explanations from other classification models, such as FFNNs, CNNs and LLMs.

8.6.4 Alternative Approaches from Related Work

Du et al. [97] propose a similar local explanation approach of deriving the contribution of each word to the final prediction based on an LSTM. However, they do not account for prediction uncertainty. Some studies show that uncertainty modeling alone can improve both the consistency and robustness of local explanations. For example, Zhao et al. [404] propose BayLIME, a Bayesian extension to the LIME framework. However, BayLIME does not explain uncertainties and is only applied to improve the consistency and robustness of relevance-based explanations. The field of uncertainty explanation within ML is usually divided into approaches that attempt to explain either the uncertainty of predictions or the uncertainty of the explanation itself. BayLUXT focuses on explaining prediction uncertainty (including aleatory and epistemic uncertainty). Thus, it aims to understand model misconceptions and unreliability.

While there are some approaches to explaining prediction uncertainty, they are mostly limited to tabular and image data. For example, Slack et al. [343] compute credible intervals for the feature importance, capturing the associated uncertainty. However, they rely on data perturbations that are difficult to apply to text classification. Others, such as Gosiewska and Biecek [122] suggest measuring the stability of local explanations between bootstrap samples to quantify uncertainty. Antoran et al. [18] propose CLUE, an approach to detect which input feature can be changed to reduce the uncertainty of an ML model. Their approach relies on searching for counterfactuals in the latent space of a deep generative model.

To our knowledge, BayLUXT is the first framework to explain the uncertainties of an ML-based text classifier, while our human evaluation is the first assessment of the effect of uncertainty explanations on human users in the field of text classification.

8.7 Conclusion

This chapter has introduced BayLUXT, one of the first XAI frameworks to explain the prediction uncertainties of RNN-based text classifiers. BayLUXT uses Bayesian statistics combined with a sequence modeling technique to decompose the aleatory and epistemic uncertainties of ML models. We have conducted a series of experiments and a human evaluation to demonstrate the correctness and appropriateness of our approach. BayLUXT does not require additional training effort over Bayesian approximation, nor does it compromise in classification performance. We briefly summarize the main findings.

- BayLUXT is suitable for extracting five types of explanations, including word- and sequence-level approximations of class probabilities, word and sequence uncertainty, and directed uncertainty explanations (DUX).

- Sequence relevances are able to highlight changes in the classification decision when an input text is analyzed from beginning to end. For example, they can indicate where in a text its overall sentiment changes from positive to negative and vice versa.

- Uncertainty explanations are a valuable source of information. We demonstrated that 85.7% of our participants stated that uncertainty explanations are either more helpful or equally helpful compared to traditional relevance-based explanations in understanding prediction results.

- Uncertainty explanations complement word relevance in enhancing the understanding of predictions. We found that 92.9% of our participants would prefer to have access to uncertainty-based explanations alongside word relevance to enrich their comprehension of artificial decision-making.

Part III

Synopsis

Chapter 9

REM - A HiL Tool for the Efficient Moderation of User Comments

Publication. This chapter is based on the 2021 paper "REM: Efficient Semi-Automated Real-Time Moderation of Online Forums" [17]. My contribution consists of the design and implementation of the tool. In addition, I took the lead in developing the HiL workflow, conducting the ML experiments, analyzing the results, and writing the research paper.

Contribution. We introduce REM, a novel HiL tool for the semi-automated real-time moderation of large-scale online forums. REM effectively saves human effort during the moderation process by combining multiple HiL design patterns and visual data analysis. REM implements the human-resource-aware Active Moderation framework we introduced in Chapter 6 to enable a highly human-resource-aware and simultaneously highly accurate moderation process. Additionally, REM provides a rich uncertainty-aware visual interactive interface to facilitate the analysis and filtering of user comments through exploratory data analysis (Visual Interactive Labeling pattern). Additionally, REM offers local explanations, such as BayLUXT (Chapter 8), to support the moderation process. Built on top of a big data architecture, REM is designed to be highly scalable and to enable real-time moderation. A preliminary experiment with domain-specific data indicates that the moderation process of REM is able to improve the classification performance of a hate speech and offensive language classifier from 78.48% to 96.08% with a manual effort of 25%.

9.1 Motivation

Online forums have become an integral part of many domains to facilitate user participation and deliberation [246]. Having recognized the vast potential of user comments, many attempts have been made to better exploit their constructive value [234, 283, 307, 311]. However, the quality of user comments varies widely [40, 285, 297]. On the one hand, studies promote the automated detection of desirable and high-quality user comments [283], such as meta-comments [138]. High-quality comments are those that are most interesting and worth reading by others, for instance, by providing a wide range of viewpoints and insights on a particular topic. On the other hand, online forums are increasingly confronted with inappropriate and toxic content such as hate-speech [80, 198] and spam [67, 247]. Quandt [297] even refers to "dark participation" to describe negative, harassing, or profoundly obscure forms of online engagement that include personal threats, insults, and malicious efforts.

Poor quality and harmful user comments considerably damage the reputation of forum providers [99, 283] and discourage user engagement. In the worst case, users turn of from online services completely potentially leading to financial losses. Typically, forum providers seek to avoid being associated with disruptive and uncivil content. Ethical and legal guidelines promote civilized interactions and put pressure on forum providers to ensure appropriate and netiquette-compliant communication. Content moderation plays a critical role in enforcing such guidelines, thereby encouraging continued user participation [274].

Content moderation in online forums is a typical classification task. It refers to "*the governance mechanisms that structure participation in a community to facilitate cooperation and prevent abuse*" [126]. In other words, forum moderation is about identifying and filtering out unwanted comments from public online discussions. Some research also refers to forum moderation as gatekeeping [40]. There are many possible reasons why a particular comment should be removed. These include comments that contain hateful, discriminatory, abusive, profane, or obscene language [274, 327], content that is irrelevant to the discussion (off-topic) [377], personal attacks or insults [301], spam or commercial solicitation [67], radicalism [275], trolling [54], fake content [247], and false or misleading information [338].

Filtering out content increases the moderation overhead for service providers [234]. The sheer amount of data makes manual moderation impractical. ML-based moderation solutions are promising, as they are cheap and fast compared to manual analysis. However, humans are very robust and adaptive compared to ML models. ML-based text classifiers generally cannot handle various situations and are unable to replace humans in real-world applications completely. Drastic measures have been taken to improve the quality of online forums, ranging from deleting individual comments, blocking certain users to shutting down the forum entirely [40, 111, 297]. However, restricting access to online forums is contrary to taking advantage of their enormous constructive potential. Helping to analyze and moderate online forums without restricting them is considered a major challenge for forum providers [90].

Tool support is needed to accurately classify and manage large amounts of user comments within online forums while maintaining high applicability. Previously, we proposed HiL as a promising approach to achieve these goals. While text classification is mostly generic, typical objectives such as high classification performance are orthogonal to the actual application domain. Thus, a tool that achieves highly accurate text classification with HiL support can address many use-cases.

9.2 System Description

In this section, we provide an overview of REM and its application to the task of moderating online forums in the domain of online journalism. We discuss the implemented HiL patterns and describe the moderation pipeline.

9.2.1 Usage of REM in the Domain of Online Journalism

The REM system is domain-independent and can be easily adapted to other datasets and use-cases, such as software engineering or social media analysis. For this purpose, the filterable dimensions, the amount of displayed information or the classification objective can be easily adapted. We consider the use-case of REM to facilitate the moderation of forums in the field of online journalism.

Online journalism is the practice of gathering, editing, and publishing news stories via the Internet. It involves delivering information to a global audience through various digital media, including websites, blogs, social media, and video-sharing sites. Online forums are an integral part of online journalism. They provide a platform where readers can interact, discuss, and debate news topics and issues, bringing a clear positive value to news organizations [234]. From a journalistic perspective, forums are a valuable source of information that can be used to receive feedback on stories, obtain new perspectives and personal stories, gather new ideas for further articles, as well as to get hints pointing out errors and typos [234]. This kind of feedback would otherwise be difficult to obtain. Reading and responding to user comments has become an essential part of the work of online journalists [63].

News organizations generally provide guidelines and rules for users to follow when participating in their online forums. These are designed to ensure that all users have a positive and respectful experience when participating in online conversations. They are intended to encourage users to interact with others in a considerate and respectful manner, while also legitimizing the removal of inappropriate content. Ethical guidelines, moderation policies, or legal constraints are typically employed to guide moderation decisions. Forum moderators are tasked with enforcing the rules of the forum. For example, *Zeit Online*[1] – one of Germany's largest news organizations – has the following rules regarding unwanted language[2]:

- "Insults have no place in discussions. If you disagree with an article or comment, criticize the content, not the author."
- "Discrimination and defamation of other users and social groups based on religion, origin, nationality, disability, income, sexual orientation, age or gender is expressly prohibited."

Spiegel Online[3], another major German news organization, is also proposing similar rules[4]:

- "Comments that are illegal, pornographic, extremist, racist, abusive, damaging to reputation or business, or inciting to commit a criminal offense will be deleted without justification."
- "Comments that contain defamation, other criminal content, advertising or commercial content will also be removed."

Recently, large online forums have received up to 10,000 comments per day [134]. The effective and applicable moderation of online forums is an open research challenge that we aim to address with REM.

9.2.2 HiL Components

REM implements several facets of the HiL approach to render the moderation process highly effective and efficient. First, REM aims to maximize the efficiency of the moderation process by implementing the Active Moderation pattern. In particular, we apply the human-resource-aware Active Moderation framework introduced in Chapter 6. Second, uncertainty quantification techniques are generally unable to detect all misclassifications. Especially those that the classifier mistakenly assumes to be correct with a high degree of certainty (i.e., unknown

[1] https://www.zeit.de/index
[2] https://www.zeit.de/administratives/2010-03/netiquette/seite-2
[3] https://www.spiegel.de/
[4] https://www.spiegel.de/panorama/netiquette-a-785155.html

unknowns [26]). Therefore, REM also relies on exploratory visualization and data analysis concepts [184]. Implementing the Visual Interactive Labeling pattern allows for the user-centered moderation of any instance. Third, REM also implements the Continuous Learning pattern, where the model is frequently re-trained as additional labeled instances become available. Over time, moderator feedback is used to re-train and potentially improve the underlying model. Fourth, in order to build user trust in the decision-making of REM, it provides local explanations such as BayLUXT for each classification outcome. Users can plug in the approach they want to use for both classification and explanation. Finally, REM allows human moderators to define the amount of effort they are willing to spend and provides guidelines on how much human involvement is actually required to achieve a specific level of classification performance. REM enables the formulation of customized moderation strategies to better meet the needs of news organizations.

Through a variety of visualizations, moderators can view and explore the current state of forum discussions. This includes which topics, articles, or users are most prone to inappropriate content, or which articles receive the most comments. In addition to actively requesting moderation in some cases, comments can be interactively filtered and individually selected for moderation. We assume that moderators will be able to extract useful information from the visualizations to identify outliers or anomalies that require special attention. By enabling user-centric moderation strategies, other types of machine errors and misconceptions can potentially be detected and corrected by humans. In particular, moderators should be able to identify toxic users or controversial topics that are prone to rudeness and derailment and require special attention. Focusing on data exploration allows moderators to better know and understand their data, ultimately leading to a process of knowledge discovery [184]. For example, moderators gain deeper insight into the dynamics of the debates. The gained knowledge about the data contributes to the effectiveness of further interactions. Overall, REM aims to enhance the quality of forum moderation by combining the strengths of human visual perception and reasoning with the computational power of ML models.

To the best of our knowledge, REM is the first HiL tool that explicitly uses the expected model behavior evaluated on a representative dataset to provide guidance on how much manual effort is required to achieve the desired level of classification performance. We outline the following domain-independent design goals for REM:

- Support for highly accurate HiL-based moderation of text classifiers (**accuracy**).

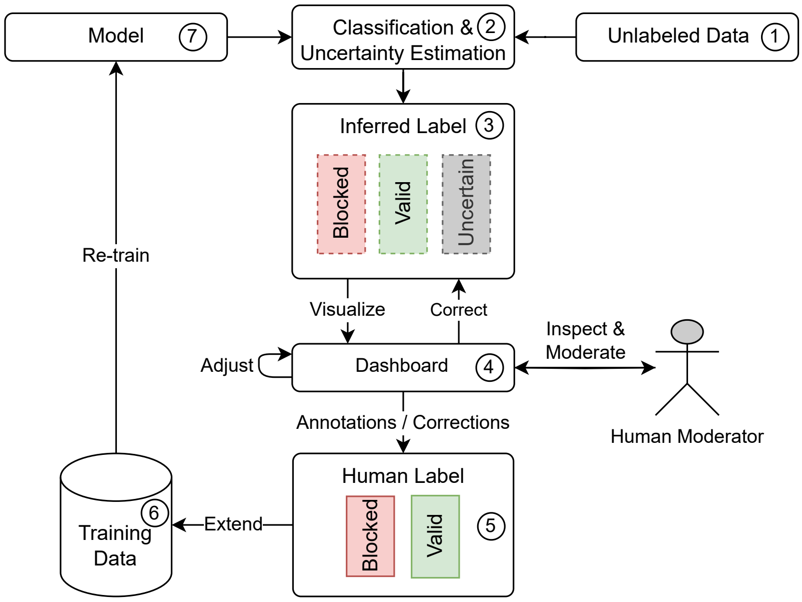


Figure 9.1: Human-in-the-Loop workflow of REM.

- Cost-effective use of human resources (**human efficiency**).
- Enable the moderation of large-scale text corpora (**scalability**).
- Support real-time moderation (**real-time processing**).
- Provide a transparent and comprehensible moderation process (**transparency**).
- Facilitate the sense-making of online debates in news discussions (**sense-making**).

9.2.3 HiL Workflow

The HiL workflow implemented in REM is illustrated in Figure 9.1. Newly posted forum comments (1) undergo immediate classification and are enriched with uncertainty information (2). Our tool employs a comprehensive moderation approach based on a binary classifier. Each comment is classified as either *blocked* or *valid*. Comments may also be marked as *uncertain* (3) if their inferred labels lack the reliability needed to achieve the desired level of classification performance. Human moderators will then be asked to provide new and more reliable labels for uncertain comments (4). However, moderators are also empowered to correct false positives and false negatives that have not been identified as uncertain. Our approach also includes a Continuous Learning component. Human-labeled instances (5) are incorporated into the training data (6) and utilized to continuously re-train the model (7). Since continuous re-training after each moderation request is inefficient when moderators work in parallel, we implement Continuous Learning in batch mode [154]. The resulting incremental update of the model weights is particularly important since the classification performance of an ML model is prone to decay over time due to data shifts [260], such as statistical disparities between training and operational data.

9.3 Requirements

This section specifies the requirements of REM, which are in part the result of an informal interview conducted with Norddeutscher Rundfunk[5] (NDR). This leading German news organization provides online forums as part of its service. The NDR states that their ability to ensure moderated online discussions was overwhelmed by the sheer volume of user comments received in near real-time. Forum moderation was outlined as a critical and open challenge that they could not handle manually due to the enormous volume of data. They emphasized that the lack of highly reliable and accurate ML models hampered their ability to automate the moderation process. Skepticism was also expressed about the feasibility of fully automated moderation approaches due to the complexity and ambiguity of language. It was also noted that they did not want to completely replace human moderators with automation. To allow at least partially moderated communication within their online forums, some drastic measures were implemented. Among other steps, the commenting function was restricted in terms of time or disabled altogether for certain topics. These measures underutilize the proven potential of user feedback.

The development of REM also draws on the findings of Loosen et al. [234], which investigated features of a visual software framework for analyzing user comments. They outline key user needs, specifically focusing on data dimensions that need to be visualized to facilitate the comprehension of large volumes of user comments. Visualizations provide valuable insights from data, which play a central role within the HiL approach. Therefore, we build REM based on their findings. Further, we have identified common functionalities expected in a user-driven analysis tool from the visual analytics literature [184, 185], such as filtering and selecting comments according to all available dimensions.

9.3.1 Functional Requirements

First, we outline the functional requirements of REM. Figure 9.2 shows a use-case diagram that visualizes how different actors can interact with the REM

[5] https://www.ndr.de/

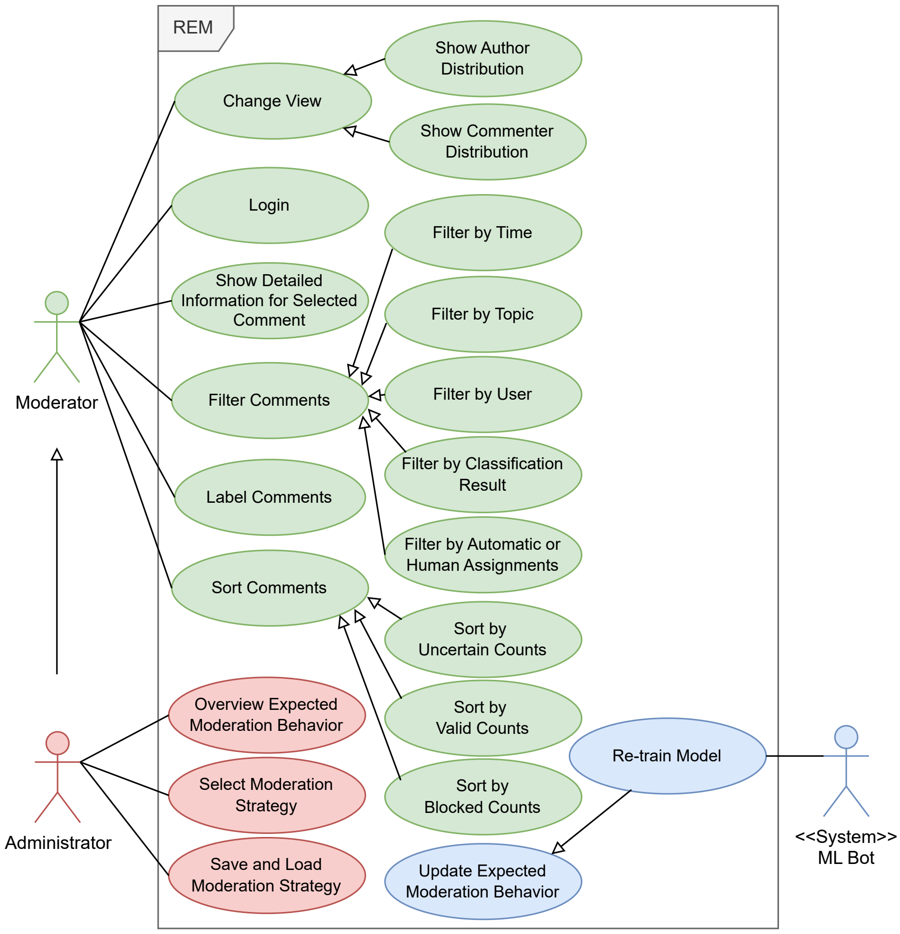


Figure 9.2: Use-case diagram of REM illustrating the interaction between users (moderators and administrators) and the system.

system, including users, administrators and other systems. We identify the following functional requirements for REM, which are specified as user stories. We further categorize the functional requirements into visual analysis (VA), ML, and HiL related requirements.

VA Related Functional Requirements. REM relies on VA, where a visual interactive interface is used to facilitate a knowledge generation process [187]. In this context, we identify the following functional requirements for REM's VA process:

FRQ-VA-1: Users should be able to overview the distribution of comments over time.

FRQ-VA-2: Users should be able to view the number of comments grouped by article, topic, commenter, and label.

FRQ-VA-3: Users should be able to filter comments by time, article, topic, commenter, and classification results.

FRQ-VA-4: Users should be able to interact directly with visual representations to filter out uninteresting comments.

FRQ-VA-5: Users should be able to select individual comments and view associated meta information.

FRQ-VA-6: User comments should be processed and visualized in near-real-time.

ML related functional requirements. REM includes an ML-based text classification pipeline. The ML part of REM should meet the following requirements:

FRQ-ML-1: All comments are classified as to whether they should be blocked or not.

FRQ-ML-2: Users should be aware of the uncertainty of any prediction.

FRQ-ML-3: Users should be informed of comments that require manual moderation.

FRQ-ML-4: Users should be able to understand why a model made classification.

FRQ-ML-5: The model should be periodically re-trained as additional training data becomes available.

FRQ-ML-6: Users should be shown the number of blocked, non-blocked, and uncertain comments.

HiL Related Functional Requirements. In order to effectively use ML models to support the moderation, several considerations must be taken into account. Human moderators need cognitive support during labelling, while minimising the amount of human supervision. For REM, the following requirements for the HiL process emerge:

FRQ-HL-1: Users should be able to make moderation decisions (block or not block comments).

FRQ-HL-2: Users can correct all ML decisions.

FRQ-HL-3: Users have access to local explanations that explain all ML-based classification decisions.

FRQ-HL-4: Users should be able to see if a human moderator has already corrected an instance.

FRQ-HL-5: Administrators should be able to select and configure a moderation strategy.

FRQ-HL-6: Administrators should be suggested the optimal saturation-based moderation strategy (Section 6.2.3).

FRQ-HL-7: Administrators should be aware of the trade-off between the amount of human labor required to achieve a desired level of classification performance.

FRQ-HL-8: Administrators should be able to judge how efficient and effective a moderation strategy is.

9.3.2 Non-functional Requirements

We summarize the non-functional requirements considered in the design and implementation of REM. Our main focus is to ensure high classification performance, scalability, security, and low model latency.

NFR-1: The ML-based classification should provide high-quality results, especially high classification performance and meaningful explanations.

The ML component of REM should maintain strong classification performance and reliable uncertainties and explanations for maximum effectiveness. The most advanced and applicable approaches to text classification should be employable.

NFR-2: The application should be compatible with all major desktop browsers.

The system must be compatible with the most popular web browsers. Google Chrome, Mozilla Firefox, Microsoft Edge, and Apple Safari in their latest versions.

NFR-3: User interaction with the system should take no more than one second.

Ensuring high user experience is crucial for HiL systems. Therefore, the system should respond within one second after the user interacts with it.

NFR-3: Visualizations should be easy to understand and handle both small and large amounts of comments.

Visualizations can easily become cluttered as the number of instances to be displayed increases. On the other hand, the tool should be used effectively in data-intensive situations (more than 10,000 texts) that exceed manual capacity.

NFR-4: Access to REM should be protected and available only to authorized users. A distinction should also be made between the roles of a moderator and administrator. Only administrators should be able to configure, select, and apply moderation strategies.

Defined user roles should secure access to REM. The default users are moderators who can view and explore the current state of all discussions. They can also perform moderation tasks. Administrators manage moderation strategies. They are responsible for determining how much effort moderation should take to achieve a desired level of performance.

9.4 Architecture

REM is designed to offer a publicly available open-source platform. We use open-source tools distributed under a free license to achieve this goal. Given the absence of a universal open-source big data framework, REM is constructed as a combination of various technologies. Our main focus is on processing speed and scalability. This section provides an overview of REM's architecture.

9.4.1 Components

The components of our tool are shown in the deployment diagram in Figure 9.3. The entry point to REM is a web application served by a Node.js[6] server, which is based on multiple micro-services. Each micro-service is built using Flask[7], a lightweight web framework for Python. The web application is constructed using Vue.js[8]. Additionally, we utilize D3.js[9], which employs a data-driven approach to DOM manipulation, to implement responsive data visualizations.

Since REM is tasked with moderating real-time data, it integrates multiple big data frameworks for stream processing. We employ Apache Kafka [203] as

[6] `https://nodejs.org/en/`
[7] `https://flask.palletsprojects.com/`
[8] `https://vue.org`
[9] `https://d3js.org/`

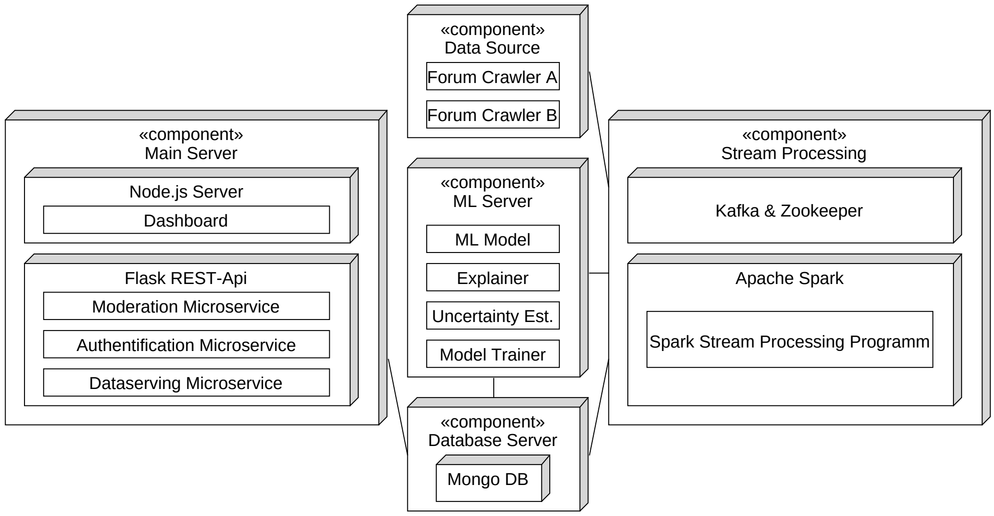


Figure 9.3: Main components of the HiL moderation tool REM.

a message broker. Apache Kafka serves as a real-time data streaming technology capable of implementing a message queuing system. It offers a scalable public-subscribe messaging pattern that acts as a middleware between one or more data sources and a stream processing platform. In this context, Kafka effectively gathers asynchronous incoming user comments and converts them into a continuous data stream. REM leverages Kafka as the central entry point for analyzing the ongoing stream of user comments. As a message queue, Kafka can receive data from any source. In an optimal setup, user comments would be directly fed into Kafka by the forum software being moderated.

To implement stream processing, REM utilizes Apache Spark's Structural Streaming API [401]. Spark Streaming is an extension of the Apache Spark [402] cluster computing ecosystem. It provides the functionality to develop an in-memory analytics platform tailored for processing real-time data. We employ Spark Streaming to handle the data stream supplied by Apache Kafka. Each stream is divided into batches and then transformed in a map-reduce fashion until the final stream of processed data is written to the serving layer. ML models can be seamlessly integrated and distributed within the stream processing pipeline.

Further, the data is persisted and served through MongoDB[10]. MongoDB is a document-based NoSQL database that is particularly suited to the needs of big data applications. MongoDB's query language relies on a map-reduce data processing pipeline. Moreover, MongoDB documents are schema-less, enabling easy extensibility and support for complex hierarchies. We chose a NoSQL database

[10] `https://www.mongodb.com/`

due to its flexible and extensible schema. Additionally, NoSQL databases offer scalability to effectively manage large datasets, making them suitable for handling diverse and expansive data scenarios.

Traditional and deep learning-based text classifiers can be used for REM. However, they need to be able to integrate uncertainty estimation and local explanations, such as BayLUXT (Chapter 8). In our implementation, the text classifier is developed using TensorFlow [1]. The initial model training is performed offline (e.g., using Proxy-based Active Learning (Chapter 7)). Model re-training is triggered by a dedicated ML service. Prior research suggests that periodically re-training a model with additional data can enhance its classification performance [20].

9.4.2 Real-time ML Pipeline

User comments from online forums are real-time data, as they occur in a continuous stream. Unwanted and toxic comments should generally be detected and removed as quickly as possible, ideally at the moment they are posted. Therefore, a moderation tool like REM must be able to classify a stream of user comments in near real-time.

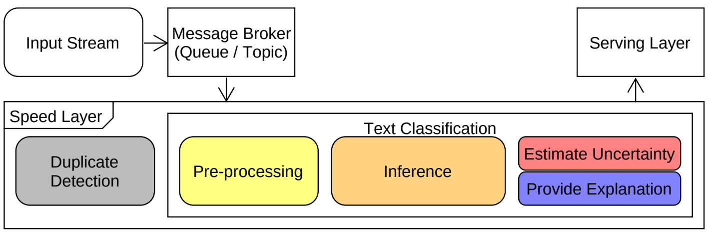


Figure 9.4: Kappa architecture of REM. The speed layer is part of the stream processing component.

Two different software architectures have been established for scalable and fault-tolerant real-time data processing: *Lambda* and *Kappa*. REM adopts the Kappa-architecture for its data processing pipeline. The Kappa-architecture, introduced by Kreps [202], is designed to process a continuous stream of real-time data. Kappa consists of two primary layers: the speed layer and the serving layer. The speed layer manages the processing of a continuous stream of data in real-time, while the serving layer focuses on delivering low-latency access to the processed data. In contrast, Lambda incorporates an additional layer for processing batches of data – the batch layer – dedicated to handling large volumes of historical data in batches. In Kappa, all data traverses through the same speed layer. REM implements Kappa because there is no need for a batch layer. Since the moderation is primarily aimed at recent data, there is no need to process large amounts of historical data. Text classifiers can be seamlessly integrated into the speed layer. Figure 9.4 illustrates our implementation of the Kappa Architecture. The speed layer encompasses the following steps:

1. **Duplicate Detection.** Comments that have been classified with the current model are not re-classified to save computational resources. Duplicate detection involves comparing the hash value of a comment text with previous texts associated with the same article and comment id.

2. **Text Classification.** Non-duplicate comments are passed through a text classification pipeline. We perform text cleaning steps, including lowercase transformation and lemmatization. The text input is then encoded and passed to a classifier for inference. For each text, the class label and its uncertainty are estimated. Further, explanations are derived. Popular ML libraries such as Keras and TensorFlow are compatible with Spark, making it easy to distribute ML models across a Spark cluster.

3. **Data Serving.** The computational results are stored and made available in the data serving layer. The serving layer is a database that provides low-latency access to the computed data and enables ad hoc queries.

9.4.3 Domain-driven Data Model

MongoDB is a document-oriented database. Entries are stored as nested JSON objects. The domain-driven data model of REM is depicted in Figure 9.5. It consists of articles (*Article*) and their associated comments (*Comment*) along with metadata, as well as users (*SystemUser*) representing the users of REM and their credentials. Furthermore, moderation strategies (*ModerationStrategy*) are incorporated, dictating which instances to label and what level of classification performance is currently being targeted.

A *Comment* entity represents a user comment within the forum and comprises associated meta information. This includes a unique identifier (*id*), the actual comment in text format, a hash of the text, and the name of the commenter. The hash is used to detect duplicate comments. As comments may be related to each other, each comment can optionally contain the *id* of the comments it references to (*parent_id*). Additionally, the number of user recommendations, such as likes, is stored. Moreover, each comment includes an object for storing its ML results. This object encompasses the class probability of blocking the comment, the inferred uncertainty, and a flag indicating whether the comment

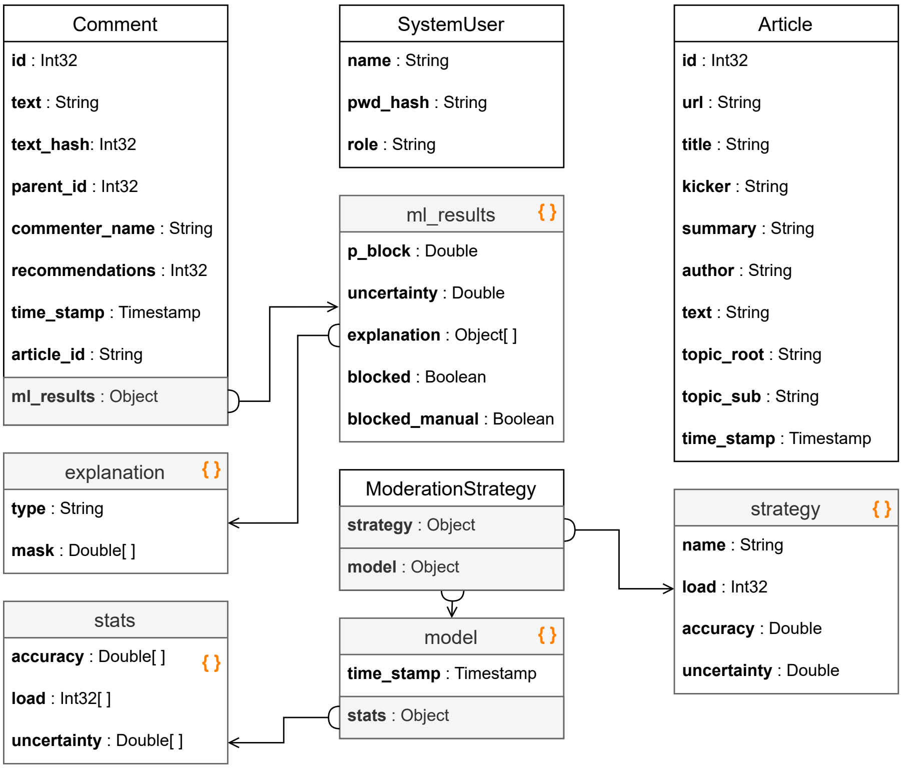


Figure 9.5: Domain-driven data model of REM. The figure illustrates the structure of individual data records.

was automatically or manually blocked. The differentiation between manual and automated blocking is necessary to facilitate a visual distinction within the tool. In addition, multiple explanations can be stored, each containing a type and an attribution mask. Furthermore, each comment includes a timestamp and a reference to the corresponding article (*article_id*).

A news page is represented by an *Article* entity. An article comprises an *id*, the article's URL, its title, the actual text, a timestamp, author names, a summary, and a kicker. The kicker is a short description of the article, typically even shorter than the summary. Additionally, in our data model, each article can belong to a two-level hierarchy of topics: a general topic (*topic_root*) and a subcategory of the general topic (*topic_sub*). Each article also has a timestamp.

The actual strategy for moderating an online forum is represented by a *ModerationStrategy* entity. Each moderation strategy consists of a strategy object and a model object. The strategy is a named uncertainty threshold derived from the expected model behavior. It also includes the anticipated classification performance and the moderation load resulting from the uncertainty threshold. An example of a strategy would be the saturation point of human-resource-aware Active Moderation. Additionally, each moderation strategy stores a model rep-

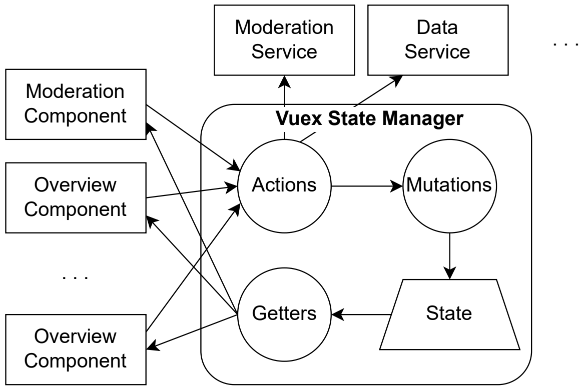


Figure 9.6: Components of the Vuex state manager.

resented by its expected behavior. Each possible moderation load is associated with its uncertainty threshold and expected classification performance.

9.4.4 Process View

This section describes the runtime behavior of REM. We focus on key features and illustrate how various processes interact using UML sequence diagrams. These diagrams show the fronted state management as well as the authentication, annotation, and moderation strategy selection processes.

Frontend State Management. REM is constructed utilizing the Vue.js web framework. To manage REM's application state and translate controller requests into REST calls, we employ Vuex[11]. The Vuex state manager is a centralized data store for all components within REM. Through Vuex, multiple views can access the same state or initiate actions consistently. This enables instant updates to all views that rely on the shared data whenever changes occur in the underlying data.

The functionality of Vuex is depicted in Figure 9.6. The state encompasses all application data. Mutations represent functions that can manipulate the state. Getters provide an abstraction layer for accessing the data. Components utilize getters to retrieve data from the state and to invoke actions, which are methods employed to trigger mutations or execute asynchronous operations such as REST calls.

Authentication. An essential requirement of REM is that only authenticated users can access the system. Consequently, all of REM's routes, services, and resources must be protected. Figure 9.7 shows the authentication process.

[11] https://vuex.vuejs.org/

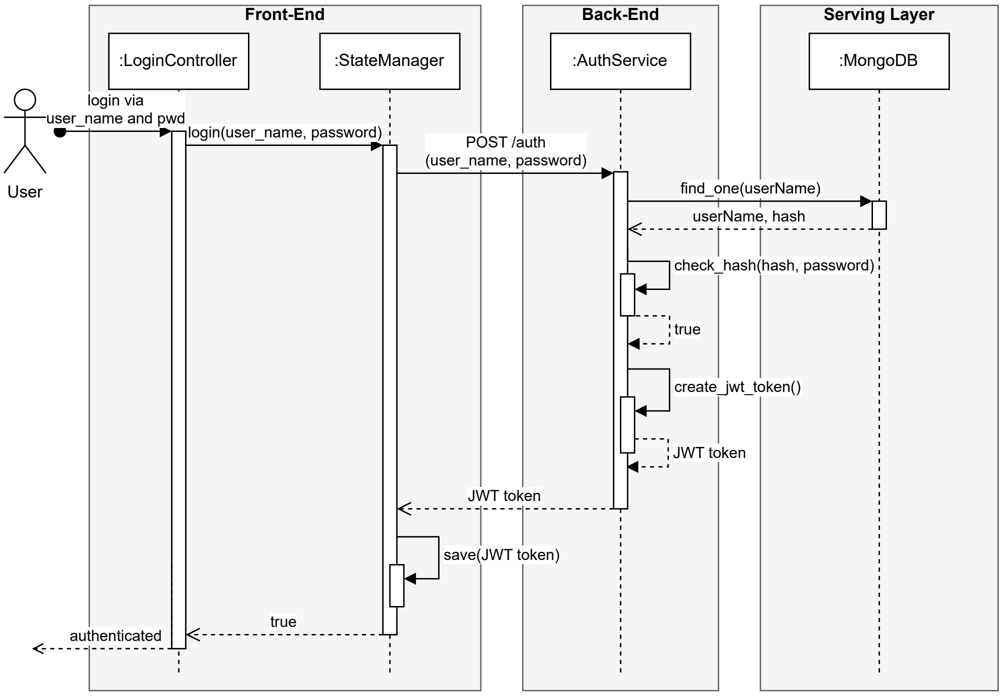


Figure 9.7: Sequence diagram of the authentication process.

First, users of REM must have valid credentials to log into the system. These are issued by privileged users and stored in the database. Authentication requires a username and password. The user must enter their credentials immediately after starting REM's web interface. The credentials are passed through the user interface to the login controller (*LoginController*). The login controller delegates the request to the state manager (*StateManager*), which manages the state of the application. The credentials are sent via a REST call to the authentication service (*AuthService*), which is responsible for ensuring the authenticity of the user. In REM, the user credentials are stored as a BSON file in a database. Instead of storing the passwords directly, only their hashes are stored. The password is preceded by a salt before it is hashed. The authentication service first looks up the username in the database and retrieves the associated hashed password. The provided password is then hashed accordingly and compared to the password hash stored in the database. If the two hashed passwords match after string comparison, a JSON Web Token (JWT) [169] is generated. The token is the result of the authentication process and is used as a signed claim that verifies the integrity and authority of the user. The JWS token is stored in the state manager and is attached to each subsequent request to authorize access. The authentication process ends when the user is authenticated and granted access to the system.

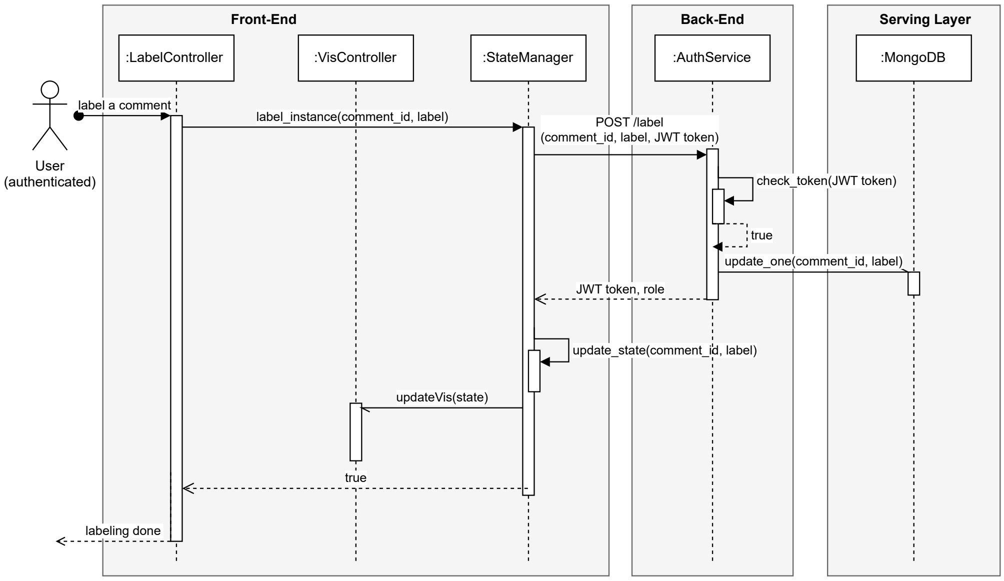


Figure 9.8: Sequence diagram of labeling a user comment.

Label Comments. Labeling is a fundamental feature of REM. The sequence diagram in Figure 9.8 outlines the labeling process. When a user provides a label, the controller (*LabelController*) captures both the *label* and the *id* of the corresponding comment. This information is then forwarded to the state manager (*StateManager*), which delegates it to the label service (*LabelService*). The *StateManager* initiates a REST call over HTTPS. If the user has the necessary credentials to access the *LabelService*, the label is changed in the database. Additionally, the *StateManager* updates the application store, triggering a local update of the visualizations in use. The label of the comment is changed and marked as manually labeled.

Apply Moderation Strategy. The sequence diagram in Figure 9.9 illustrates the process of applying a new moderation strategy, which is divided into two tasks. First, selecting a moderation strategy and observing its effects, and second, applying the strategy. To select a moderation strategy, a user must have valid credentials and the role "*administrator*".

When a moderation strategy is selected in REM, an uncertainty threshold describing the selected moderation strategy is estimated by the *ModController* and passed to the *StateManager*. As with any subsequent API call, the JWT token is validated. A database query is performed to calculate the expected change from the current moderation strategy to the new one. This includes the number of valid, blocked, and uncertain comments. The aggregations are passed to the state manager, and the values are shown to the user. When the user wants

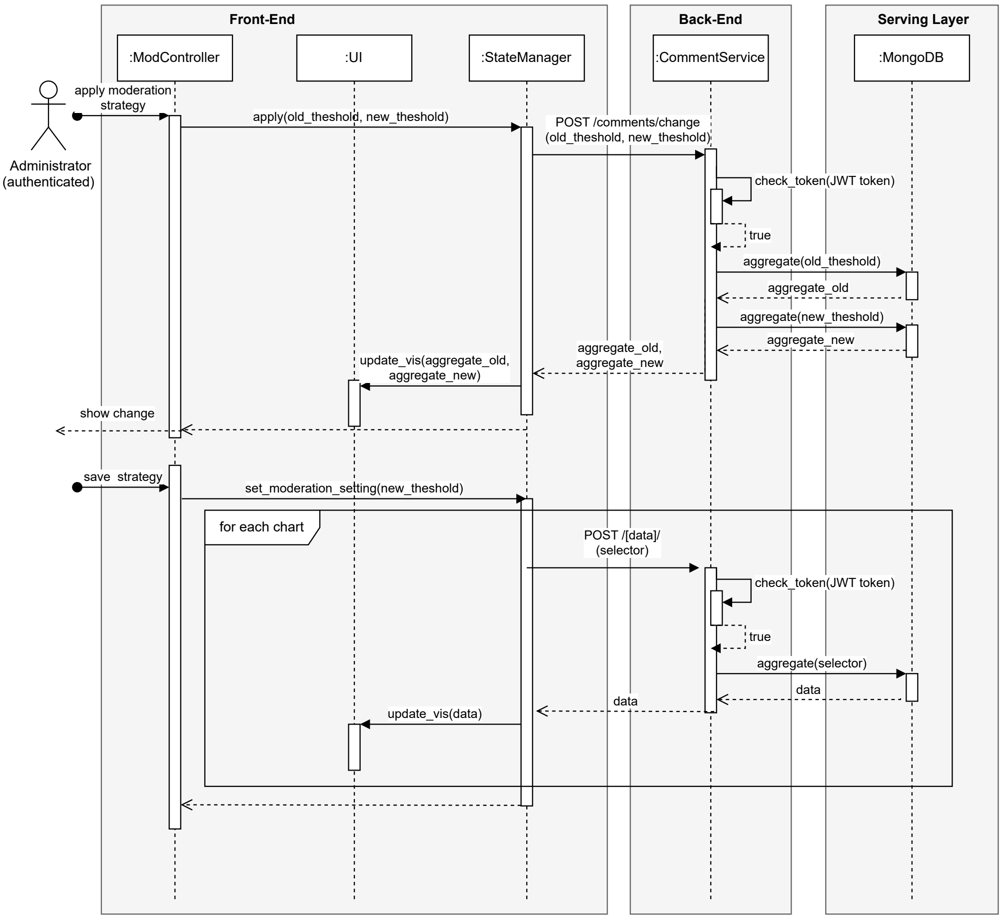


Figure 9.9: Sequence diagram of applying and saving a moderation strategy.

to commit a selected moderation strategy, he or she sends a *commit* command through REM's UI. The strategies underlying the uncertainty threshold are passed to the *StateManager*, which initiates a REST call for each visualization to be updated. On each request, new data to be visualized is retrieved from the serving layer and the visualizations are updated accordingly.

9.5 System Implementation

We will briefly describe the implementation of the user interface of REM. Figure 9.10 shows the main page of our tool. The user interface consists of three views, which are described in the following. We share the source code[12] of our prototype along with a video demonstrating the main features of the tool[13].

[12] `https://github.com/jsandersen/REM`
[13] `https://youtu.be/cA92Io_xr6Q`

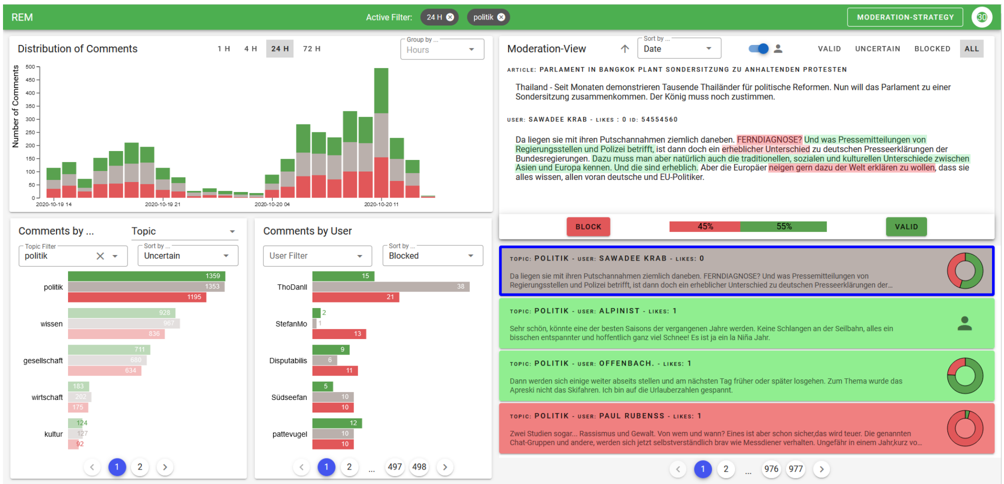


Figure 9.10: The main page of REM showing the Context-view (left) and the Moderation-view (right).

9.5.1 Context-view

The Context-view shown in Figure 9.10 provides an overview of the distribution of comments over time and journalistic entities such as topics, articles, and users (commenters). The upper bar chart shows the distribution of comments over time. The x-axis represents the time dimension, and the y-axis indicates the total number of comments. Each bar represents the number of classified comments within a given time period using a three-color scheme. Blocked comments (e.g., inappropriate, offending, etc.) are red, valid (non-blocked) comments are green, and uncertain comments are gray. Since moderating online forums is a real-time task, the tool focuses on recently posted comments. The granularity of the time dimension can be changed using the button group at the top. Possible intervals are minutes, hours, and days. Furthermore, moderators can choose to show comments from the last 72 hours, or just the last hour.

The lower part of the Context-view shows the distribution of comments concerning journalistic entities, i.e., topics such as politics or economics, and the articles identified by their titles. The second graph shows the commenting behavior of users. Each graph can be sorted by the number of uncertain, blocked, valid, and all comments. All visualizations in the Context-view are responsive to filter operations. These can be triggered by clicking on the bars. Specific entities can also be searched for using a text box. Multiple filters can be chained for flexible visual analysis.

9.5.2 Moderation-view

The Moderation-view shown in the right part of Figure 9.10 provides a detailed overview of the selected comments from the Context-view. All selected comments are listed. Each entry in the list consists of the text of the comment and additional meta information, such as the corresponding topic, the user who posted it, and the number of recommendations given by other users. Similar to the color scheme used in the Context-view, the color of each cell represents the current label of the comment. The pie chart visualizes the model's conditional class probability for blocked and valid class outcomes. For highly uncertain predictions, both class results would be nearly equal. A "human" icon is displayed instead of the pie chart when a human has already labeled a comment. The list can be filtered to show only uncertain, valid, or blocked comments. Entries can also be sorted by timestamp or uncertainty. Moderators are asked to at least moderate the uncertain user comments. Comments that have already been manually moderated can also be hidden for a better overview.

Detailed information about a selected comment is displayed at the top of the view. This includes additional information about the corresponding article, followed by the text of the comment. The view also provides class probabilities and local explanations, to facilitate humans better understand the classification decision and maintain user trust. The selected comment is highlighted with a blue box in the comment list. Actual moderation is done using the buttons at the bottom. An uncertain comment can be blocked or marked as valid. Predictions can also be corrected. Additionally, the moderator can agree on (accept as a ground truth) artificial predictions to provide more training data for re-training. Corrections and additional labels are synchronized directly with the database.

9.5.3 Control-view

Figure 9.11 depicts the Control-view, which is used to control the moderation process. It can be accessed from the *"Moderation Strategy"* button on the main page. REM implements the Active Moderation pattern. The expected classification performance of the underlying classifier when a certain number of the most uncertain predictions are manually validated is shown by the line graph. On the right, a user can select different moderation strategies, which are also highlighted in the line graph. For each moderation strategy, the expected classification performance and the required effort are shown. A user can select a predefined moderation strategy or define a custom strategy by hovering and clicking on a point in the line graph. A moderation strategy affects the number of predictions marked as uncertain. The current strategy is displayed at the top.

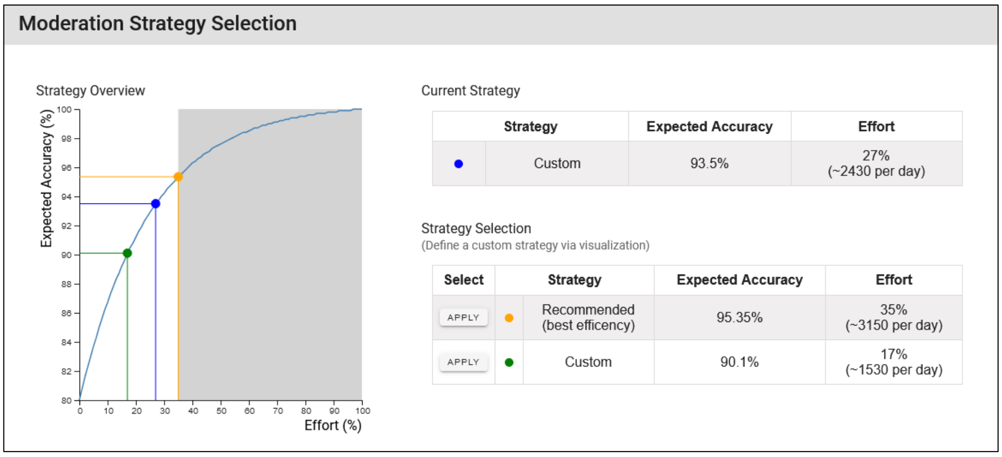


Figure 9.11: The Control-view of REM for managing the moderation strategy.

Usually only some moderators should be able to change the moderation strategy, and thus the classification performance of forum moderation. Therefore, the Control-view is secured by requiring the role “administrator”.

Because moderation efficiency decreases as workload increases, a point is reached where further moderation effort yields only marginal improvements. To inform users of such inefficiencies, REM provides a recommended moderation strategy to optimize human moderation effort in terms of gains in classification performance. We calculate the recommended moderation effort according to our human-resource-aware Active Moderation framework (Chapter 6). The gray area in the line graph highlights inefficient workloads.

9.6 Active Moderation Experiment

We conduct a preliminary experiment to demonstrate that the Active Moderation approach integrated into REM can efficiently improve the classification performance of a hate speech classifier during its operational use. Our experiment uses the dataset provided by Davidson et al. [80], which consists of 24,782 Twitter comments labeled as offensive, hate speech, or neither. In our experiment, we classify the comments into blocked (offensive and hate speech) (83.2% of the total) and valid comments (16.8% of the total). Since the data is highly imbalanced, we use the balanced accuracy [50] to measure the classification performance. We use 7,868 instances for training, 7,868 for validation, and 9,046 for testing. A workload of 100% corresponds to manually reviewing 9,046 comments, the expected daily number of comments in our application scenario. We adopted SBERT to compute the text encodings used as the input for an FFNN. We employ MCD to estimate prediction uncertainties.

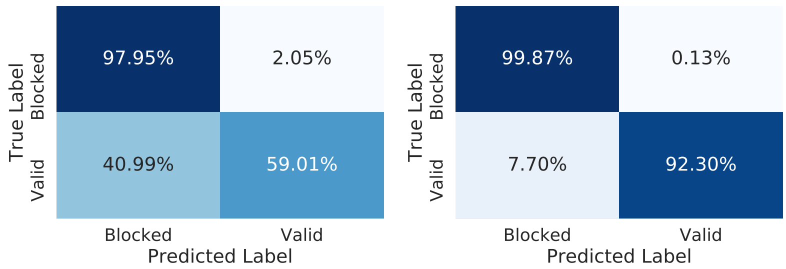


Figure 9.12: Normalized confusion matrix of the initial classifier (left) and the same classifier when 25% of the most uncertain predictions were moderated (right).

Our trained classifier achieves a balanced accuracy of 78.48% (without manual moderation). Next, we simulate manual moderation by selecting the ground truth labels. The balanced accuracy of a moderated classifier is computed based on the inferred and manually corrected labels. The results show that uncertainty-based moderation is more efficient than a random moderation strategy, where the instances to be labeled are selected randomly. For example, moderating 25% of the data based on its uncertainty leads to a balanced accuracy of 96.08%. In comparison, a random moderation strategy requires a moderation effort of 81.8% to achieve the same accuracy, and is thus much less efficient.

The confusion matrix of the initial and moderated classifiers is shown in Figure 9.12. The fully ML-based classifier obviously has difficulties in correctly identifying valid comments. Only 59.01% of the valid comments were correctly identified. By moderating only 25% of the data, the detection of valid comments can be increased to 92.30%. The Control-view depicted in Figure 9.11 illustrates that the classification performance can further be improved by increasing the amount of human involvement. Thus, our approach is able to further improve the classification performance of a text classifier with reasonable manual effort. A more general investigation of the Active Moderation pattern is stated in Chapters 5 and 6.

9.7 Comparing Explanations from BayLUXT and ChatGPT

Decoder-based LLMs such as ChatGPT provide strong performance in many NLP tasks, without requiring any training data (Prompted-base Learning). LLMs are able to classify text and explain themselves in a human-like verbalized form. In a preliminary study, we investigate how an LLM compares to BayLUXT in providing word attributes for the REM system, including word relevance and word uncertainty. Both approaches can potentially be used within REM. Our goal is to highlight the differences between the two approaches. We prompt ChatGPT to provide explanations similar to those provided by BayLUXT, and also investigate ChatGPT's rationale for providing these explanations.

We consider the task of sentiment analysis as it is done in the human evaluation. We also use the same text instances. We apply BayLUXT and ChatGPT-3.5 [279] to provide explanations of word-relevance and -uncertainty. For ChatGPT, we first translate the sentiment analysis task into a natural language prompt. We then extend the prompt to provide local explanations and confidence scores for the classification task. Explanations are queried to take the shape of a Python-like array, with each element indicating the attribution of a particular word. These arrays are then visualized using a heat-map, as in BayLUXT. We rely on a prompt from the literature Huang et al. [163] and a self-developed prompt. Given a small set of text instances, we compare the explanations obtained with those of BayLUXT. Second, we prompt ChatGPT to provide uncertainty explanations. We also experimentally investigate the ability of LLMs to further explain their explanations in natural language. Our preliminary study of BayLUXT and ChatGPT focuses on qualitative aspects.

9.7.1 Word Relevance

First, we investigate the ability of ChatGPT as a decoder-based LLM to provide self-explanations for the task of sentiment analysis. We consider a prompt from the literature, as shown in Table C.1, and a self-engineered prompt, as shown in Table C.2. We compare the results of both prompts with the corresponding explanations provided by BayLUXT.

Figure 9.13 shows the relevance explanations retrieved from ChatGPT and BayLUXT on several examples. First, it is noticeable that ChatGPT highlights significantly more words as relevant than BayLUXT. For example, the words "*a*", "*an*", "*you*", "*keep*", "*that*" and, "*will*" are highlighted as highly relevant. Also, ChatGPT tends to treat words that refer to relevant words as highly relevant. Examples include the phrases "*weak writing*", "*bad acting*", and "*thrilling story*". However, changing the phrase *weak writing* to "*weak camera work*" does not significantly change the sentiment. Masking relevant words from the input and measuring how this affects the model's confidence is a typical approach to evaluating explanations. The relevance of the masked word is expected to be proportional to the shift in the output distribution [268]. Furthermore, the trimmed variation "*An absolute disaster of a movie with weak, poor, and a predictable.*" has the same class probability as the original sentence. Thus,

Label: Confidence	Text and Explanations	Type
Positive: 0.66	It was an excellent performance by the actors and a great setting. Unfortunately, the plot was terrible. I hope that the actors find new projects.	BayLUXT
Negative: 0.85	It was an excellent performance by the actors and a great setting. Unfortunately, the plot was terrible. I hope that the actors find new projects.	ChatGPT – P1
Negative: 0.75	It was an excellent performance by the actors and a great setting. Unfortunately, the plot was terrible. I hope that the actors find new projects.	ChatGPT – P2
Negative: 0.51	An average thriller that keeps you guessing until the end, but falls short on execution.	BayLUXT
Negative: 0.7	An average thriller that keeps you guessing until the end, but falls short on execution.	ChatGPT – P1
Negative: 0.65	An average thriller that keeps you guessing until the end, but falls short on execution.	ChatGPT – P2
Negative: 1.00	An absolute disaster of a film with weak writing, poor acting, and a predictable plot.	BayLUXT
Negative: 0.9	An absolute disaster of a film with weak writing, poor acting, and a predictable plot.	ChatGPT – P1
Negative: 0.9	An absolute disaster of a film with weak writing, poor acting, and a predictable plot.	ChatGPT – P2
Positive: 0.90	A gripping tale of love, betrayal, and redemption that will leave you emotionally invested till the very end.	BayLUXT
Positive: 0.95	A gripping tale of love, betrayal, and redemption that will leave you emotionally invested till the very end.	ChatGPT – P1
Positive: 0.95	A gripping tale of love, betrayal, and redemption that will leave you emotionally invested till the very end.	ChatGPT – P2

Figure 9.13: Word relevance extracted by BayLUXT compared to ChatGPT using two prompts (P1 and P2).

ChatGPT is prone to mark words as relevant which do not contribute to the actual classification outcome. As ChatGPT, BayLUXT considers the words that describe an emotion to be relevant, such as "*weak*", "*poor*", and "*excellent*". As outlined in Chapter 8, BayLUXT aims to only highlight words that contribute to the actual class outcome.

When comparing the two prompts, there are only minor differences that present some inconsistencies. Both prompts offer very similar confidence scores. The only noticeable difference is that P2 considers the word "*thriller*" as positive and "*betrayal*" as negative, similar to BayLUXT. In both cases, P1 considers these words to have the opposite sentiments.

9.7.2 Word Uncertainties

Next, we prompt ChatGPT to provide explanations of word uncertainties similar to those provided by BayLUXT. Since there is no comparable study in the literature, we adapted P2 to query for word uncertainties instead of word rele-

Confidence/ Uncertainty	Text and Explanations	Type
0.66 / 0.40	It was an excellent performance by the actors and a great setting. Unfortunately, the plot was terrible. I hope that the actors find new projects.	BayLUXT
0.75 / 0.65	It was an excellent performance by the actors and a great setting. Unfortunately, the plot was terrible. I hope that the actors find new projects.	ChatGPT – P3
0.51 / 0.50	An average thriller that keeps you guessing until the end, but falls short on execution.	BayLUXT
0.80 / 0.60	An average thriller that keeps you guessing until the end but falls short on execution.	ChatGPT – P3
1.00 / 0.00	An absolute disaster of a film with weak writing, poor acting, and a predictable plot.	BayLUXT
0.90 / 0.80	An absolute disaster of a film with weak writing, poor acting, and a predictable plot.	ChatGPT – P3
0.90 / 0.14	A gripping tale of love, betrayal, and redemption that will leave you emotionally invested till the very end.	BayLUXT
0.85 / 0.70	A gripping tale of love, betrayal, and redemption that will leave you emotionally invested till the very end.	ChatGPT – P3

Figure 9.14: Word uncertainty extracted by BayLUXT compared to ChatGPT.

vances. The prompt is shown in Table C.3. In particular, we asked how much each word contributed to the overall uncertainty of the classification decision. Similar to BayLUXT, we asked not only for words that added uncertainty, but also for words that reduced uncertainty.

Figure 9.14 shows the explanations obtained. The results indicate that the explanations of ChatGPT can be interpreted similarly to those of BayLUXT, but also show significant differences. For the first text, both explanations are very similar. In both cases, the words "*excellent*" and "*great*" are considered certain. Also in both cases the term "*Unfortunately*" contributes uncertainty to the prediction. Both approaches consider words in the first sentence as adding certainty, while words in the second sentence add uncertainty to the prediction. In contrast, ChatGPT highlights many more words. Comparing the highlighted words with the word relevance in Figure 9.13, it can be seen that both explanations follow the same scheme of highlighting the first sequence of positive words as certain and the following uncertain words as uncertain. The second sentence follows a similar scheme, where the first relevant word is considered to contribute uncertainty, while the relevant word of the other class contributes uncertainty. While BayLUXT is maximally confident without any uncertainty in the third example, ChatGPT reports a high confidence but also a high uncertainty score. Surprisingly, many words are considered to add uncertainty, while no words reduce uncertainty, which is contrary to the behavior of BayLUXT. In the forth sentence, BayLUXT provides a class label with no uncertainty, while ChatGPT reports high uncertainties across all highly influential words. In addition, the sentence receives the highest overall uncertainty.

Further, we did not find a direct correlation between the uncertainty and confidence scores provided by ChatGPT. A prediction with a confidence of 0.9 received more than twice the uncertainty of a prediction with a confidence of 0.8. Further, a low confidence does not necessarily cause a higher uncertainty. Compared to BayLUXT, ChatGPT is much more inconsistent in its confidence and uncertainty scores. For example, non-relevant words can also contribute to the prediction uncertainty.

9.7.3 Explaining Explanations

ChatGPT has the advantage of being able to answer follow-up questions. For example, ChatCPT can provide textual explanations of its predictions by simply prompting for additional explanations. This includes asking why certain words are relevant, not relevant, or more relevant than others. BayLUXT, on the other hand, obviously cannot provide any additional information.

Our experiments show that ChatGPT provides plausible explanations. For instance, to the question *"Why is the word 'acting' negative?"*, ChatGPT answers: *"[...] When words are in close proximity to other negative words or phrases in a sentence, they can inherit the negative sentiment associated with those words through semantic association or contextual interpretation. In this case, the presence of "poor acting" in the sentence likely influenced the classification of "acting" as negative."*. Similarly, ChatGPT is able to explain word uncertainties such as "*The word 'Unfortunately' contributes to uncertainty in the sentiment classification because it introduces a conflicting sentiment within the review.*"

However, natural language explanations clearly lack precision, which leads to a certain degree of vagueness. For example, when ChatGPT is asked why some words are uncertain, it states: "*There might still be slight uncertainty due to the potential for different interpretations or nuances in the context.*" Furthermore, we notice that in some cases, ChatGPT reconsiders and corrects its reasoning when asking for explanations regarding specificity attribution scores. In particular, ChatGPT tries to improve the results according to the user's questions. In our experiments, we were able to change almost all attribution scores with a prompt similar to the one shown in Figure 9.15.

9.7.4 Discussion

We have compared the explanations of BayLUXT and ChatGPT, which differ significantly in the way they receive explanations. BayLUXT extracts relevance and uncertainty explanations based on the logits of a supervised learned RNN.

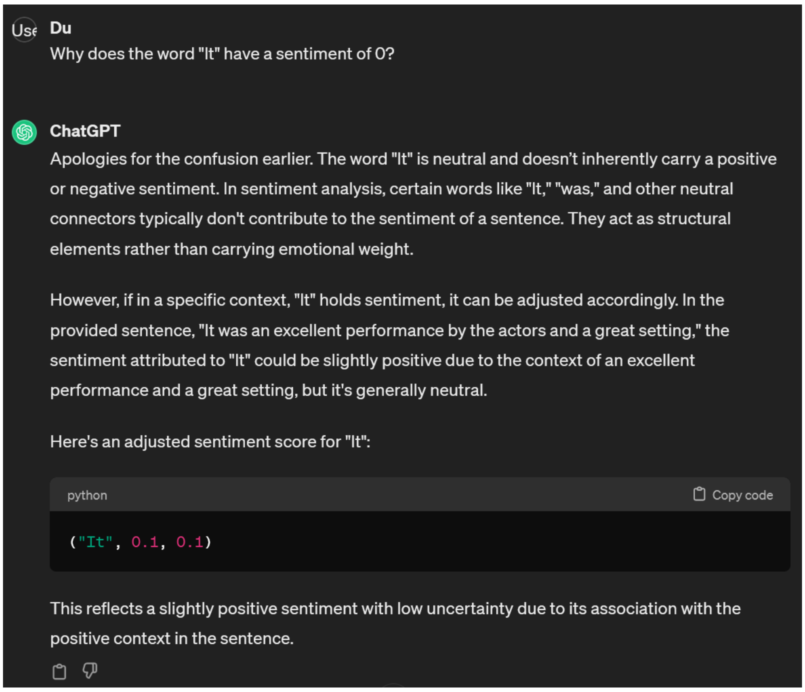


Figure 9.15: ChatGPT altering an attribution after prompting for an explanation.

The explanations are derived directly from the weights, allowing statistical approaches to uncertainty estimation. ChatGPT is a decoder-based LLM hidden behind a back-box API. It does not provide access to its internal weights. Word relevance and uncertainties are the response to a prompt, so they are estimated as a masked language problem.

Our results show that ChatGPT is well suited for providing local explanations. While it is challenging to judge which explanation is better, ChatGPT can well express how it solves a text classification task. However, ChatGPT does not provide intuitive confidence and uncertainty scores. Nevertheless, it functions well for text classification and is able to express reasonable explanations.

ChatGPT is built on top of the Transformers architecture, which is based solely on the attention mechanism. This serves as a plausible explanation for ChatGPT's behavior. The attention mechanism allows the contextual meaning of each word to influence the meaning of the others. It is conceivable that the internal embeddings of nouns incorporate the meanings from their corresponding adjectives. Thus, both words in the phrase 'excellent performance' are associated with a positive sentiment. Therefore, ChatGPT's explanations are not considerably better or worse than those of BayLUXT. Both reflect how the different models arrive at their classification results.

Ultimately, users will have to decide which approach to classification, coupled with their way of explaining predictions, makes more sense to them and which they want to use within REM.

9.8 Related Tools from the Literature

Previous HiL tools have attempted to facilitate text classification. Most tools focus on the task of interactive model building, also known as Active Learning [332], and rely heavily on multidimensional projections [100]. Similar to REM, HiL annotation tools typically offer a visual-interactive interface to guide human involvement [34, 153]. Neves and Ševa [270] provide a comprehensive overview of document annotation tools.

Seifert and Granitzer [328] present a basic user-centered Active Learning tool in which humans sequentially select and label instances for the next training iteration. Similar to our tool, the authors use prediction uncertainty to guide human involvement. However, they do not integrate an explorative analysis component. Heimerl et al. [147] propose a user-centered visual-interactive Active Learning tool for text documents. Annotators can re-train models in batches and inspect statistics about the model's performance. However, the authors do not consider uncertainty thresholds. The tool provided by Höferlin et al. [153] allow annotators to directly manipulate the underlying model. This approach requires the annotators to be experts in ML, which is not the case for typical forum moderators in the domain of online journalism. The HiL labeling tool proposed by Choi et al. [69] facilitates an attention mechanism to explain predictions to annotators. They aim to reduce the time needed for annotation decisions and increase labeling efficiency. Their results could improve our tool.

Link et al. [229] introduce a similar semi-automated process for moderating social media content. In addition to relying on prediction uncertainty, they define untrustworthy sources that require additional attention. Similar to our tool, human moderation is requested when a prediction does not meet a certain confidence level. In contrast, they do not focus on optimizing moderation to achieve a desired level of classification performance and human effort. Riehle et al. [310] propose a platform for semi-automated moderation of online discussions. Similar to our tool, comments are automatically pre-moderated, and human moderators can correct or agree with the predicted labels. However, moderators are not guided to identify comments that require manual attention, nor do they evaluate the effectiveness of the moderation process. Forum 4.0 [134] is a HiL tool for classifying, aggregating, and visualization of user comments. Domain experts can train ML models by labeling data instances according to their needs. Classifiers are built on the fly through an interactive labeling process with continuously monitored validation metrics. However, they have not implemented the Active Moderation pattern and rely only on a single visualization.

9.9 Discussion

The HiL system REM can improve the reliability of moderation decisions while still relying heavily on ML-based text classification. Incorporating human expertise and input helps to identify and correct errors, fill knowledge gaps, and align tool outputs with real-world scenarios and user requirements. This human involvement facilitates customization and adaptation of the tool to specific application needs or user preferences. By incorporating domain-specific knowledge, the tool's behavior can be tailored to different contexts, resulting in more personalized and effective outcomes. Bringing people into the loop also promotes trust and transparency within the system. Users gain a clearer understanding of how the tool works and their ability to provide input and feedback fosters a sense of control and understanding. As a result, user acceptance and confidence in the tool's results are increased. Another benefit of the HiL approach is the ability to address biases and ethical considerations in the results generated. Human annotators or users can identify and correct biased or harmful results, promoting fairness and responsible AI practices. In addition, the interaction between the ML tool and human users promotes collaborative learning and knowledge sharing. Users can contribute insights, corrections, or alternative perspectives, ultimately improving the overall classification performance and learning of the tool. This collaboration creates a symbiotic relationship between humans and the tool, facilitating mutual learning and growth.

9.10 Conclusion

This chapter has presented REM, a novel tool for the semi-automated real-time moderation of large scale online forums. REM implements several HiL patterns to meet the requirements of accurately moderating large-scale online forums. Forum moderators contribute to REM by providing expert knowledge about the problem being solved. No technical ML knowledge is required.

REM has been designed specifically for online journalism. The growing demand for online participation and the increasing number of user comments raise the challenge of filtering out harmful and unwanted content from public debates in online forums. Since manual moderation does not scale well and fully automated approaches often lack the required level of classification performance, we propose a semi-automated moderation approach. Our approach maximizes the efficiency of manual efforts by targeting only those comments that require human intervention (Active Moderation). Our tool provides a rich visual interactive environment for exploring online debates. We conduct a preliminary experiment to demonstrate the suitability of the moderation approach implemented in REM and publicly release its source code. Since REM can potentially be used with different classification models, their form of explanations may also differ. In a preliminary study, we compared the explanations provided by BayLUXT with some provided by ChatGPT. While we found significant differences in the way they explain themselves, the user must ultimately decide which one best suits their needs.

REM combines HiL and visual analytics methods to enable an efficient and more accurate moderation process. We implement the human-resource-aware Active Moderation framework to reduce and optimize human effort by building on the prediction uncertainty of models. We also present a rich uncertainty-aware visual-interactive interface to facilitate moderation through exploratory data analysis. Built on a big data architecture, our tool is highly scalable and enables real-time moderation. A preliminary experiment has shown that our moderation approach is able to improve the classification performance of a hate and offensive language classifier from 78.48% to 96.08% by moderating only 25% of the data.

REM can be adapted to more general use-cases where annotators need to efficiently improve the classification performance of binary classifiers while taking advantage of Continues Learning. Future work should focus on evaluating our tool in terms of user experience, acceptance, and usefulness in supporting the moderation of online forums.

Chapter 10

Conclusion

This chapter summarizes the contributions of this thesis, discusses the results, and outlines its limitations.

10.1 Summary of the Contributions

The main contribution of this thesis is sixfold. First, we provide a catalog of 12 HiL design patterns for software engineers to better understand and design HiL systems. Second, we comprehensively investigate the effectiveness of the uncertainty-based moderation of text classifiers, called Active Moderation, to improve their classification performance during operation. Third, we propose and study a saturation-based moderation strategy to effectively limit human resources during Active Moderation while maximizing the classification performance of text classifiers. Fourth, we adopt a Proxy-based Active Learning framework for text classification to maintain highly applicable training in terms of high user experience, human cost-effectiveness, and high classification performance. Fifth, we propose and investigate the suitability of one of the first frameworks based on Bayesian statistics to explain the prediction uncertainty of text classifiers, which we refer to as BayLUXT. Finally, we present REM, a functional prototype for the effective real-time moderation of online forums that integrates the previous contributions. Taken together, these contributions provide valuable insights into how the HiL approach can facilitate the applicability of text classification systems. Below is a detailed summary of our contributions and findings:

HiL Pattern Catalog. We have identified a general lack of design knowledge and best practices for developing HiL systems. To address this, we have conducted a literature review of current HiL approaches, covering a total of the 200 most relevant records from Google Scholar and the Google search engine. We have considered both research papers and gray literature.

We found that most existing HiL approaches are concerned with the cost-effective use of human labor during the initial training phase. Also, many approaches rely on human assistance to discover and prevent unreliable predictions during operation. In total, we have identified 12 different HiL approaches. Based on these findings and our experiments, we developed a HiL pattern catalog that covers best practices for HiL model training and operational use. Since the scope of this thesis focuses on domain expert feedback, we have concentrated on HiL approaches that do not require technical feedback, i.e., no direct model building and hyperparameter selection.

From these findings, we have extracted 12 HiL design patterns, including seven training and five operation patterns. For each pattern, we outline its goal, human role, structure, advantages, and potential challenges. Furthermore, we outline when the patterns should be applied and refer to existing implementations from the literature. A common goal of HiL training patterns is the cost-effective human labeling of training data, while operation patterns are mainly aimed at improving the quality of decision-making. While our pattern catalog is by no means complete, it is a step towards facilitating the design and deployment of HiL systems.

Computational-aware Active Moderation. In Chapter 5, we have investigated the potential of the Active Moderation pattern to increase the applicability of ML-based text classifiers in situations where highly complex but accurate classifiers are not applicable due to environmental concerns or a lack of computational resources. For this purpose, we have investigate the suitability of the prediction uncertainties of lightweight classifiers for Active Moderation and evaluate their effectiveness.

We found that the probability distributions between incorrect and correct classifications differ significantly when using lightweight classifiers, potentially allowing for computationally efficient Active Moderation. For app reviews, the mean probability of all incorrect and correct classifications was 66.4% and 89.42%, respectively. Next, we showed that removing the most uncertain instances from the test dataset (i.e., 0%, 10%, 20%, and 30%) significantly increases its overall F1 score. For example, the F1 score of an MLP* increased from 83.71% to 93.02% when the classification performance was calculated using only the 70% most certain predictions. Since Active Moderation allocates the most uncertain predictions to human annotators, we then investigated the combined semi-automated classification performance. In particular, how much human effort is required to achieve a given F1 score. We observed significant improvements in the F1 score when the most uncertain predictions were manually corrected. For example, on the News dataset, when 9.48% of the most uncertain predictions were manually corrected, the F1 score of the MLP* improved from the original 89.28% to 95%. As the most uncertain predictions are assigned to human annotators, we next investigate the combined semi-automated classification performance of the ML model and humans. In particular, we have investigated how much human effort is required to achieve a specific F1 score. In our experiments, the F1 score increased from an initial F1 score of about 78% to at least 95% in five out of six datasets by manually validating less than 29.6% of the data. Since human annotators can make mistakes, we also accounted for some level of human noise. We found that even a 15% noisy human within Active Moderation can still produce significantly more accurate results than a purely automated classifier with a reasonable amount of manual effort. Finally, we have investigated the runtime behavior of the classifiers on a weak infrastructure (only 4 GB of main memory) to assess the applicability in resource-constrained settings. We found that several of the best performing classifiers, such as LR and MLP, require less than one second to infer the labels of classifying around 15,000 instances.

Human-resource-aware Active Moderation. In Chapter 6, we have conducted a second study of the Active Moderation pattern. We have focused on the problem that human effort is costly and scarce. To be applicable, human effort must be limited to instances where human corrections are actually needed. We have proposed a human-resource-aware Active Moderation framework based on a saturation-based moderation strategy to minimize human effort while maximizing classification performance. Our experiments included several deep learning-based text classifiers, including a state-of-the-art DistilBERT model, as well as several uncertainty estimation techniques and datasets.

Without human intervention, we achieved F1 scores of up to 94.1%, which may still be too low for practical use-cases. When applying Active Moderation to deep learning models, we found that human efficiency (the number of moderation requests that are misclassifications and require human intervention) steadily declines as moderation effort increases. In particular, we observed human efficiencies of 50%, 41%, and 32% when moderating 1%, 10%, and 20% of the data, respectively. This underscores the need to limit moderation to a certain amount to avoid wasting human effort. Therefore, we proposed a saturation-based moderation strategy that estimates the natural point of saturation for the moderation on a test dataset to limit human involvement. We show that our saturation-based moderation strategy can effectively limit human effort while still achieving very high F1 scores. In particular, using DistilBERT, we achieved an F1 score of at least 98% on all datasets with a manual load of $\sim 25\%$. When humans are noisy, we found that F1 scores of about 98%, 97%, and 96% are still achieved with noise levels of 5%, 10%, and 15%, respectively. Overall, our experiments show that our moderation framework can significantly improve the F1 score and thus the applicability of both weak (CNN) and strong (DistilBERT) deep learning-based text classifiers.

Proxy-based Active Learning. In Chapter 7, we have addressed the lack of user experience when adopting Active Learning to train deep learning-based text classifiers. The use of state-of-the-art ML models is highly desirable due to their expected high classification performance. However, state-of-the-art models are usually very slow during training and inference due to their high complexity. While Active Learning promises to reduce human effort during the initial labeling of training data, it requires frequent model re-training, resulting in lengthy interruptions and waiting times. This makes Active Learning with highly complex models impractical from a user experience perspective. We have investigated a Proxy-based Active Learning framework where a lightweight, low-latency classifier is used within Active Learning. The collected training data is then used to train an additional state-of-the-art model, which is then used during operation. In this context, we have investigated the performance of BERT when trained on multiple Proxy-sampled datasets. We have examined Logistic Regression (LR) and FastText as Proxies, using a label budget of 300 and 500 instances.

We found that data collected by a Proxy can significantly improve the F1 score of a BERT classifier compared to sampling random training data. After 500 Active Learning iterations, the relative improvement in micro F1 score ranges from 1.75% to 3.18%. The macro F1 score increased by 2.54% to 15.30%. Furthermore, we found that training an additional BERT classifier on the Proxy-sampled data can improve the micro F1 score by up to 7.27% and the macro F1 score by up to 19.34% compared to the direct operational use of the Proxy. We have additionally investigated the runtime behavior of the Proxies. We show that LR is significantly faster than FastText. In particular, FastText requires 6.81 to 8.32 seconds to perform training and inference, while LR requires only 0.36 to 0.68 seconds to perform the 500th Active Learning iteration. Thus, Proxy-based Active Learning is suitable for fast user interactions of less than 1 second. Finally, we have compared the imbalance of the Proxy-sampled training data to the randomly sampled data. We found that the Proxy-sampled data is much more balanced and thus considered to be of higher quality, which is also reflected in the increased classification performance. Overall, we show that Proxy-based Active Learning is sufficient to improve the classification performance of strong classifiers when they cannot be used directly within Active Learning. Our framework mitigates user experience concerns while enabling Active Learning in computationally constrained training environments.

BayLUXT: Bayesian Local Uncertainty Explanations. In Chapter 8, we have proposed BayLUXT, a framework for explaining potential uncertainties in text classification outcomes. BayLUXT is specifically designed to explain RNNs such as LSTMs. It applies a decomposition technique that reveals how much each word contributes to the overall classification uncertainty and class activation score. BayLUXT provides five different types of explanations: word relevance, word uncertainty, sequence relevance, sequence uncertainty, and directed uncertainty explanations (DUX). We have conducted several experiments to assess the correctness and appropriateness of BayLUXT. Our investigation focuses on the task of sentiment analysis.

First, we have demonstrated the correctness of BayLUXT through several experiments. We show that BayLUXT is able to identify transitions within a text between sentiment changes. Furthermore, we found that the most relevant words for positive and negative sentiment are reasonable, supporting the appropriateness of our framework. We also found a relationship between relevance scores and uncertainty. Overall, we show that BayLUXT is sufficient to extract meaningful words for both positive and negative sentiment and to provide plausible uncertainty scores.

Second, we have conducted a human evaluation that contained several comprehension tasks, including the explanations provided by BayLUXT, and a questionnaire. We found that uncertainty explanations provide a valuable source of insight into the inner workings of a text classifier. Uncertainty explanations help human users to better understand a model's prediction. In our study, 92% of respondents stated that uncertainty should always be explained along with word relevance.

REM. In Chapter 9, we have introduced REM, a visual interactive HiL tool for the efficient moderation of online forums. REM is built on top of multiple big data frameworks, including Apache Kafka, Apache Spark, and MongoDB. It also includes a scalable real-time ML pipeline. REM implements several HiL patterns, including Active Moderation, Visual Interactive Labeling, and Continuous Learning. In addition, a visual-interactive user interface allows for knowledge discovery of the forum while enabling user-centric labeling. The requirements were derived from an interview with NDR and the literature. We present REM's domain-driven data models, outline its architecture, and present its user interface and workflows.

The concept of prediction uncertainty plays an important role in REM. Instances are not only classified into "*blocked*" or "*valid*", but may also be flagged as uncertain. This would indicate the need for human supervision. Humans can define an uncertainty threshold, which influences the number of instances to be considered uncertain. In addition to user-defined moderation strategies, REM also suggests the saturation-based moderation strategy. We have conducted an empirical evaluation of REM on hate speech data. We show that moderating 25% of the most uncertain data increased the balanced accuracy from an initial 78,48% to 96.08%. Further, we have conducted a preliminary investigation of two explanation techniques compatible with REM: our BayLUXT framework and ChatGPT. Both approaches provide reasonable explanations, but also show substantial differences. Practitioners can incorporate their preferred explanation techniques into the REM tool.

10.2 Threats to Validity

We summarize the threads to validity from the previous chapters and place them in the overall context of this thesis.

HiL Pattern Catalog. Our proposed pattern catalog is not intended to be complete, and we may have overlooked some established HiL patterns. During the analysis, we limited our scope to the 200 most relevant records in the literature across two search engines. We did not consider further literature due to the high screening effort. Moreover, HiL is a highly heterogeneous concept. Existing HiL approaches may not explicitly identify themselves as such, making it difficult to identify them using our search query alone. In addition, the HiL research domain employs various terminologies that cannot be comprehensively captured in a single search query, complicating efforts to gain a holistic view of the HiL approach.

Furthermore, our pattern catalog incorporates our experience in designing HiL systems, which introduces subjectivity into the catalog. Additionally, there is a bias towards text classification, as it serves as the primary objective of our work. Other ML domains may encounter different challenges and tasks that we may not have considered in our analysis.

Investigations of HiL Frameworks. The HiL experiments conducted in Chapters 5, 6, and 7, lack to some extent in generalizability. As with any empirical

ML experiment, we had to make some compromises that affected the validity of the results. For practical reasons, only a limited number of datasets, model architectures, and configurations can be considered. We may have missed a configuration that would have produced better results. To mitigate these limitations, we considered a variety of well-performing concepts, such as commonly used classification models or established uncertainty modeling techniques. We have also focused on a variety of real-world datasets covering software engineering, online journalism, and social media analytics. There is no guarantee that other datasets will not yield significantly different results. Furthermore, all of our experiments involve only English datasets, which limits their generalizability to other languages. Languages vary greatly in structure, grammar, and vocabulary, and what works well for English may not be directly applicable to other languages. Furthermore, English-centric datasets may contain socio-cultural biases. To mitigate this limitation, we have included multiple datasets from different domains that cover a variety of jargons. Further research is needed to evaluate other languages accordingly.

Another limitation concerns the label quality of the ground truth datasets. Most of the considered datasets are labeled by humans, who may make mistakes. Furthermore, some labels are automatically derived from manually assigned information, such as star ratings, which are highly subjective and may not match the actual text. Although we rely on established datasets, errors and noise are not excluded. Noise in the data can corrupt the classification performance and make it difficult to assess whether the model is actually making a mistake. We also simulate human effort by selecting the ground truth class label for each moderation or labeling request. This is done because we do not have the resources to have several thousand user comments manually labeled by domain experts. We follow the common practice of considering feedback from domain experts in their domain to be somewhat reliable. Using fully labeled datasets allowed us to simulate human labelers. Since it is unrealistic to assume that humans do not make mistakes, we also simulate human guessing by randomly selecting labels. However, it is unrealistic to assume that humans make random mistakes without taking into account the characteristics of the actual text. To mitigate this problem, we compare different noise levels across multiple datasets.

Our results also do not emphasize that the proposed HiL frameworks are clearly domain-independent. Although we covered three domains, each with several different real-world datasets, we mainly considered comparatively short texts that fit into the feature space of the classifiers. Domains with much longer text or statistical differences to our datasets may perform differently. Dealing with overlong sentences may require additional pre-processing steps, such as feature averaging, that need to be considered.

Regarding the external threads, the anticipated affordable amount of human effort may still be too large for some use-cases. While the HiL approach significantly improves classification performance or reduces human effort during training, manual data annotation is still very expensive and only available in limited quantities. It is up to the user to decide if the effort is worth it or if it is really necessary to benefit from HiL. We also ran our experiments with fully labeled datasets. Thus, the random splits in the training, test, and evaluation data are all from the same distribution. Real-world data could change over time, which would negatively affect the classification performance.

Explaining Prediction Uncertainties. Our investigation of BayLUXT is limited in scope, which affects the generalizability of our findings. First, we consider only one dataset covering the task of sentiment analysis. Furthermore, we randomly selected a few samples to demonstrate the technical suitability of the explanations provided. The applicability to other tasks or domains is unknown. We have focused on a task that is fairly easy to understand and does not require domain-specific knowledge that is difficult to understand for a broad audience.

In addition, our human evaluation is limited because we only had 15 participants. Also, we could only evaluate a small sample of explanations to keep the evaluation time short. Furthermore, we only consider DUX in the human evaluation. It remains unknown whether other uncertainty explanations would have performed differently. Furthermore, our participants were students who were familiar with the development of ML systems and the application of XAI techniques. It is possible that other samples of people would have produced different results. To mitigate this, we compared our framework to the baseline of traditional relevance explanations.

REM. REM is easily adaptable to domains other than online journalism, such as facilitating the HiL-based classification of app store data or software issues. However, we have not reviewed and analyzed the specific requirements for these applications. It may be that slightly different views are needed to optimally support these domains. Furthermore, the development of REM is based on an informal interview at a German news organization and requirements from the literature. It is possible that other news organizations have different requirements for their moderation processes, and thus we cannot generalize to all news organizations. Further research is needed to test the user acceptance of REM with actual end users.

10.3 Further Work

We suggest directions for future work.

User Experiments with Domain Experts. One potential area for further investigation is conducting experiments investigating the user acceptance of the proposed HiL approaches. While the empirical results are promising, research is needed to investigate whether and to what extent domain experts are willing to be in the ML loop. A promising use-case is to facilitate the daily work of practitioners, such as forum moderators or software engineers. Further research should focus on conducting applicability tests with real end users.

HiL with Decoder-based LLMs. While we focus on supervised ML models for text classification, decoder-based LLMs provide a highly promising alternative to solve text classification tasks. However, these models have significant differences, such as not revealing internal parameters nor providing softmax probabilities. While confidence and uncertainty scores can be extracted as part of the reasoning task, their extraction poses uncertainties that remain opaque. Further research can investigate the suitability of their self-expressed prediction uncertainty to implement Active Moderation or other operational HiL patterns. Also, decoder-based LLMs offer novel ways of explaining themselves, such as textual justifications, which attain less attention to facilitate the HiL approach.

To further reduce the bottleneck of human effort in HiL systems, a recent idea is an *LLM-in-the-Loop* [77] paradigm. Since decoder-based LLMs have shown promising results in replicating human-like behavior, they can potentially eliminate some of the human effort through automation. Further research could investigate this paradigm for its efficiency and effectiveness in text classification.

Investigation of other HiL Patterns for Text Classification. While we promote a catalog of 12 HiL design patterns, our investigation focuses on cost-efficient implementations of Active Moderation and Active Learning. Further work could be dedicated to providing cost-effective and applicable implementations of other patterns accordingly. This would highly contribute to the research areas of HiL text classification, i.e., human-resource-aware Safeguards.

REM Improvements and Extensions. We could also improve the Visual Interactive Labeling interface of REM. A promising extension would be the Semi-supervised Interactive Labeling pattern. This would allow the moderator to easily inspect semantically similar user comments that are likely to share the same label. Further, we could improve on REMs' Continuous Learning integration. An extension could monitor and communicate the improvements achieved through re-training, providing an incentive for humans to stay in the loop. Additionally, we could improve REM by considering the time required to perform the moderation as well. Yet, REM considers each moderation request to cost the same amount of human resources. Differentiating the complexity or length of user comments could improve the moderation throughput in human time-constrained settings.

Furthermore, online forums also increasingly consist of images. We could extend REM with image classification to detect inappropriate images. Further research could investigate the adaptability of our human-resource-aware Active Moderation framework to image data.

Transfer of REM to other Domains. While REM is specifically designed to facilitate the efficient moderation of large-scale online forums, it can be adapted to other domains accordingly. Further research could investigate the transfer and use of REM for alternative use-cases, such as classifying app store reviews and issue types. In these cases, the classification object is different and covers more than two classes.

Adaption of Uncertainty Explanations. With BayLUXT, we propose one of the first investigations of uncertainty explanations for text classification. While our findings clearly promote the value of uncertainty explanations for human users, they play a minor role in the current XAI literature. Further research is needed to adapt BayLUXT to other NN architectures, such as CNNs or LLMs, or even to provide a model-agnostic approach. Also, more extensive user studies covering multiple domains and actual domain experts contribute to the field of uncertainty explanation.

Part IV

Appendices

Appendix A

Benchmark Results

A.1 Computational-aware Active Moderation

Classifier	App Store 0%	App Store 10%	App Store 20%	App Store 30%	News 0%	News 10%	News 20%	News 30%	Hate Speech 0%	Hate Speech 10%	Hate Speech 20%	Hate Speech 30%
DT	56.74	56.58	56.30	56.26	61.29	62.23	62.21	62.33	63.12	63.22	63.22	63.13
		-0.27	-0.50	-0.07		1.52	-0.03	+0.20		+0.16	+0.0	-0.14
RF	65.40	66.97	66.60	64.33	78.64	82.72	86.78	90.11	68.40	66.21	62.69	58.96
		+2.41	-0.54	-3.41		+5.19	+4.91	+3.84		-3.20	-5.31	-5.95
kNN	69.73	72.72	74.61	76.03	76.23	80.99	84.77	88.36	57.51	53.53	53.30	53.45
		+4.30	+2.60	+1.90		+6.25	+4.66	+4.25		-6.93	-0.41	+0.28
GNB	63.69	66.34	68.83	71.41	76.34	80.00	83.71	86.55	69.94	72.44	74.48	76.82
		+4.16	+3.75	+3.75		+4.79	+4.64	+3.40		+03.57	+2.82	+3.14
SVM	74.22	77.79	80.76	83.73	83.23	88.11	92.11	95.44	76.70	79.00	80.98	82.59
		+4.81	+3.82	+3.68		+5.86	+4.55	+3.61		+3.00	+2.51	+1.99
LR	77.01	81.13	84.84	88.19	85.13	90.42	94.52	97.11	78.17	81.08	83.63	85.57
		+5.35	+4.57	+3.95		+6.22	+4.53	+2.74		+3.72	+3.15	+2.31
MLP	77.55	81.68	85.70	89.53	85.36	90.63	94.73	97.34	78.25	81.40	84.03	86.15
		+5.33	+4.93	+4.47		+6.17	+4.53	+2.75		+4.01	+3.23	+2.53
MLP*	78.08	82.05	85.80	88.96	86.16	91.30	95.35	97.63	79.19	82.61	85.40	87.82
		+5.08	+4.57	+3.69		+5.97	+4.43	+2.39		+4.32	+3.38	+2.83
B-MLP*	77.96	81.97	85.53	88.88	86.03	91.18	95.18	97.56	79.10	82.46	85.18	87.43
		+5.13	+4.35	+3.91		+5.98	+4.38	+2.51		+4.25	+3.30	+2.63
AVG	71.15	74.14	76.55	78.59	79.82	84.18	87.71	90.27	72.26	73.55	74.77	75.77

Classifier	Issues 0%	Issues 10%	Issues 20%	Issues 30%	Reuters 0%	Reuters 10%	Reuters 20%	Reuters 30%	TREC 0%	TREC 10%	TREC 20%	TREC 30%
DT	48.96	49.01	48.89	48.85	50.23	50.28	49.99	50.18	46.02	45.81	45.72	45.42
		+0.09	-0.24	-0.08		+0.10	-0.58	+0.39		-0.45	-0.21	-0.65
RF	63.44	65.74	67.78	69.93	72.89	77.75	82.16	82.47	67.71	69.97	72.11	73.96
		+3.61	+3.11	+3.17		+6.67	+5.68	+0.37		+3.33	+3.06	+2.57
kNN	62.02	64.21	66.24	68.19	73.94	79.45	82.91	83.02	72.26	74.80	76.96	79.36
		+3.53	+3.17	+2.93		+7.46	+4.36	+0.13		+3.52	+2.88	+3.12
GNB	60.42	62.38	64.01	65.58	68.64	74.66	80.53	84.96	61.55	63.90	66.10	68.33
		+3.25	+2.62	+2.45		+8.77	+7.86	+5.49		+3.82	+3.44	+3.37
SVM	67.71	70.44	72.52	74.70	87.26	93.67	96.82	98.43	83.60	87.48	89.74	92.38
		+4.03	+2.95	+3.02		+7.34	+3.36	+1.66		+4.65	+2.58	+2.94
LR	68.38	71.09	73.42	75.84	87.70	94.60	97.41	98.86	84.92	88.42	90.75	92.90
		+3.96	+3.28	+3.30		+7.87	+2.97	+1.49		+4.13	+2.63	+2.36
MLP	67.92	70.49	72.92	75.13	86.81	94.17	97.30	98.44	86.39	90.31	93.13	94.36
		+3.77	+3.45	+3.04		+8.48	+3.32	+1.17		+4.53	+3.13	+1.31
MLP*	69.27	72.02	74.48	76.97	87.48	94.97	97.69	98.95	86.72	90.23	92.21	93.61
		+3.97	+3.41	+3.35		+8.56	+2.86	+1.28		+4.05	+2.19	+1.52
B-MLP*	69.31	72.05	74.48	76.82	87.27	94.51	97.75	99.15	86.76	89.72	92.03	93.52
		+3.96	+3.37	+3.14		+8.29	+3.43	+1.43		+3.40	+2.57	+1.63
AVG	64.16	66.38	68.30	70.22	78.02	83.78	86.95	88.27	75.10	77.85	79.86	81.54

Table A.1: F1 scores of different classifiers after removing a certain proportion of the most uncertain predictions from the test dataset, employing BERT embeddings. The table also shows the relative changes from the previously removed portion. The two best and worst F1 scores for each setting are highlighted in green and red, respectively.

	App Store				**News**				**Hate Speech**				
Classifier	89%	91%	.93%	95%	91%	93%	95%	97%	89%	91%	93%	95%	
DT	74.87	79.16	83.51	88.45	76.34	81.48	86.80	92.16	70.30	75.83	81.26	86.57	0% human noise
RF	36.70	41.40	47.12	53.47	21.49	26.78	32.96	42.37	28.78	34.15	41.16	50.23	
kNN	31.23	36.26	40.58	47.25	25.40	30.95	37.10	44.74	30.37	35.72	42.66	51.92	
GNB	43.02	47.68	52.25	58.20	28.09	34.41	41.44	53.22	46.34	54.36	62.65	72.53	
SVM	23.65	27.78	32.23	37.80	12.69	17.10	22.79	28.92	19.06	23.56	29.57	38.17	
LR	18.49	22.65	27.82	33.64	8.89	12.79	17.30	23.46	17.72	22.48	28.53	37.25	
MLP	17.80	21.68	25.75	31.57	8.55	12.37	16.94	23.30	17.82	22.99	29.62	38.46	
MLP*	16.74	20.65	25.53	30.63	7.47	11.25	15.68	21.44	16.04	20.90	26.96	35.01	
B-MLP*	17.08	21.25	25.53	30.91	7.61	11.52	16.14	21.65	15.99	20.71	26.56	34.34	
DT	82.01	87.23	91.90	97.12	84.74	90.92	96.78	-	77.10	83.13	89.03	95.06	5% human noise
RF	42.02	48.19	55.54	63.20	24.02	30.57	40.70	63.64	32.04	39.00	48.77	64.92	
kNN	35.67	40.52	47.78	54.66	28.42	35.42	44.15	59.21	33.89	41.06	51.25	68.77	
GNB	47.87	53.10	59.98	69.96	32.07	40.32	55.09	-	53.11	62.90	74.83	90.00	
SVM	26.03	31.10	36.86	47.87	13.91	19.09	26.08	37.63	20.73	26.21	34.57	50.07	
LR	20.18	24.81	31.32	40.93	9.63	14.05	19.43	29.80	19.29	25.00	33.40	46.56	
MLP	19.40	23.62	29.41	38.58	9.27	13.53	18.98	28.81	19.45	25.76	34.64	49.41	
MLP*	18.02	22.50	28.19	35.92	8.07	12.46	17.43	26.71	17.58	23.23	31.16	45.05	
B-MLP*	18.68	23.19	28.69	35.95	8.30	12.76	18.07	26.94	17.50	23.12	30.77	43.38	
DT	91.49	97.47	-	-	97.10	-	-	-	85.17	92.06	99.03	-	10% human noise
RF	47.28	55.60	64.99	-	27.60	37.59	-	-	36.87	47.80	71.12	-	
kNN	39.92	48.00	56.66	-	33.18	42.30	-	-	38.46	50.23	78.78	-	
GNB	53.88	61.83	-	-	38.06	-	-	-	62.02	7.57	98.13	-	
SVM	29.72	35.70	49.62	-	15.32	22.16	31.52	-	22.84	30.16	45.42	-	
LR	22.81	29.66	40.77	-	10.43	15.40	22.66	-	21.43	28.74	41.92	-	
MLP	21.59	26.72	35.54	-	10.01	15.10	22.24	-	21.87	30.20	45.30	-	
MLP*	20.59	26.66	34.11	-	8.69	13.66	20.20	-	19.41	26.89	39.73	-	
B-MLP*	21.50	26.94	34.73	-	8.81	14.21	20.69	-	19.13	26.07	37.72	-	
DT	-	-	-	-	-	-	-	-	97.33	-	-	-	15% human noise
RF	59.67	-	-	-	33.81	-	-	-	47.37	-	-	-	
kNN	48.03	-	-	-	41.36	-	-	-	49.54	-	-	-	
GNB	70.62	-	-	-	-	-	-	-	87.02	-	-	-	
SVM	33.26	54.01	-	-	17.30	26.32	-	-	26.07	39.70	-	-	
LR	25.03	35.29	-	-	11.48	17.59	-	-	24.42	38.24	-	-	
MLP	23.28	31.20	-	-	10.93	17.01	-	-	24.93	39.20	-	-	
MLP*	22.65	31.54	-	-	9.57	15.00	32.39	-	22.02	32.54	-	-	
B-MLP*	22.90	31.13	-	-	9.74	15.62	-	-	22.10	31.88	-	-	

Table A.2: F1 score of Active Moderation utilizing BERT encodings on the App Store, News, and Hate Speech datasetss. Each cell denotes the percentage of the most uncertain classification results that require manual annotation to attain a specified F1 score at a particular level of human noise. F1 scores that are unattainable due to high levels of human misclassification are denoted by “-”. The two best and worst F1 scores for each setting are highlighted in green and red, respectively.

Classifier	Issues				Reuters				TREC				
	79%	81%	83%	85%	93%	95%	97%	99%	89%	91%	93%	95%	
DT	58.97	62.92	66.89	70.76	86.35	89.92	93.85	97.79	79.67	83.33	87.10	90.46	0% human noise
RF	29.93	34.46	39.18	44.00	17.44	21.45	28.95	44.86	42.94	48.82	56.72	67.44	
kNN	32.32	36.75	41.63	46.54	17.65	21.11	25.93	38.36	38.84	45.53	53.76	61.46	
GNB	36.07	40.64	45.67	50.99	27.12	31.27	36.78	49.69	62.60	69.96	77.42	83.74	
SVM	22.35	26.79	31.66	36.78	5.69	8.92	13.93	24.36	9.91	16.10	22.24	28.66	
LR	20.61	25.06	29.68	34.60	4.50	7.66	13.05	22.54	8.33	13.74	20.09	26.85	
MLP	21.07	25.51	30.44	35.50	5.55	8.52	13.19	23.84	9.11	13.17	19.35	25.87	
MLP*	19.26	23.65	28.34	33.04	4.85	8.06	12.10	21.96	7.56	12.87	18.51	26.38	
B-MLP*	19.42	23.97	28.61	33.42	5.25	7.99	12.03	21.04	8.30	14.01	19.52	27.25	
DT	63.20	67.40	71.54	75.84	-	-	-	-	90.96	95.63	99.83	-	5% human noise
RF	32.44	37.46	42.66	47.98	21.27	39.26	-	-	56.89	73.52	96.77	-	
kNN	34.99	39.98	45.53	50.58	21.22	29.21	-	-	38.84	45.53	53.76	61.46	
GNB	39.19	44.50	50.14	56.49	33.29	-	-	-	78.43	88.51	97.11	-	
SVM	24.29	29.26	34.64	40.28	6.62	11.84	-	-	13.17	21.37	31.45	-	
LR	22.31	27.22	32.50	38.02	5.02	9.87	20.76	-	10.25	19.25	29.03	-	
MLP	22.83	27.96	33.34	38.75	6.32	10.77	20.78	-	11.66	17.37	26.28	56.45	
MLP*	20.87	25.76	30.86	36.02	5.87	9.52	18.06	-	9.95	17.07	28.09	48.96	
B-MLP*	21.03	26.14	31.30	36.74	6.04	9.80	18.55	-	11.09	18.65	29.47	54.91	
DT	67.85	72.38	76.90	81.43	-	-	-	-	-	-	-	-	10% human noise
RF	35.17	40.78	46.68	52.92	-	-	-	-	-	-	-	-	
kNN	38.02	44.20	49.62	56.26	-	-	-	-	-	-	-	-	
GNB	42.88	48.98	55.88	63.30	-	-	-	-	-	-	-	-	
SVM	26.26	32.19	38.24	44.63	8.08	-	-	-	17.00	26.41	-	-	
LR	24.19	29.73	35.59	41.98	6.04	-	-	-	11.66	23.42	-	-	
MLP	24.75	30.68	36.58	42.67	7.41	-	-	-	12.20	21.91	-	-	
MLP*	22.57	28.02	33.56	39.44	7.27	13.47	-	-	11.32	21.88	-	-	
B-MLP*	22.89	28.40	34.24	40.36	7.22	13.47	-	-	12.00	25.74	-	-	
DT	73.80	78.63	83.59	88.38	-	-	-	-	-	-	-	-	15% human noise
RF	38.70	45.03	51.96	59.47	-	-	-	-	-	-	-	-	
kNN	42.17	48.56	55.91	64.68	-	-	-	-	-	-	-	-	
GNB	47.88	55.82	64.50	72.91	-	-	-	-	-	-	-	-	
SVM	28.88	35.54	42.84	50.81	-	-	-	-	32.49	-	-	-	
LR	26.39	32.65	39.49	46.99	11.93	-	-	-	25.03	-	-	-	
MLP	27.00	33.68	40.36	47.88	12.05	-	-	-	21.40	-	-	-	
MLP*	24.51	30.54	36.85	44.08	9.84	-	-	-	20.19	-	-	-	
B-MLP*	25.18	31.35	38.14	45.53	11.07	-	-	-	25.71	-	-	-	

Table A.3: F1 score of Active Moderation with BERT encodings on the Issues, Reuters, and TREC datasets. Each cell denotes the percentage of the most uncertain classification results that require manual annotation to attain a specified F1 score at a particular level of human noise. F1 scores that are unattainable due to high levels of human misclassification are denoted by "-". The two best and worst F1 scores for each setting are highlighted in green and red, respectively.

A.2 Human-resource-aware Active Moderation

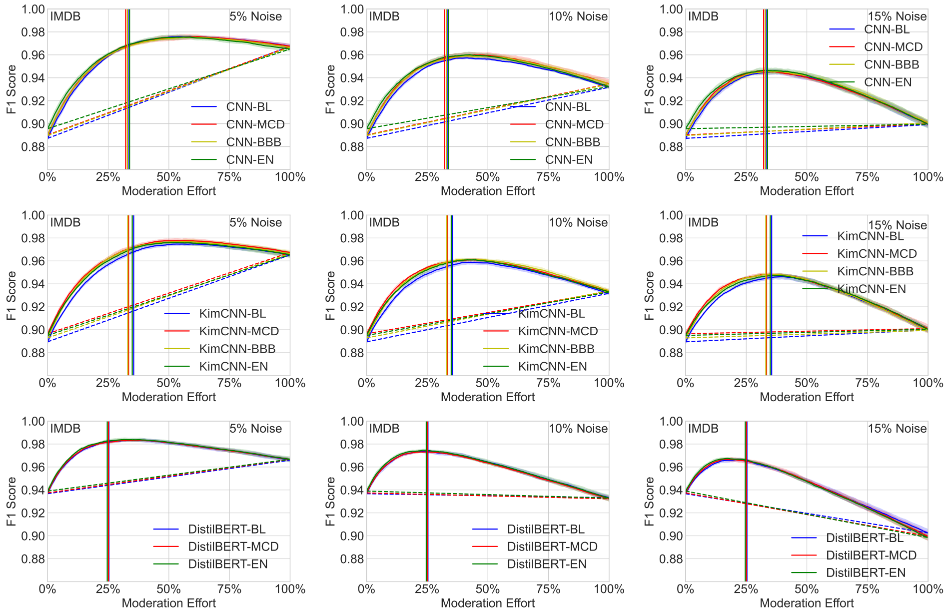


Figure A.1: Active Moderation under human noise (5%, 10%, and 15%) on the **IMDB** dataset using various uncertainty modeling techniques. Vertical lines illustrate the saturation points.

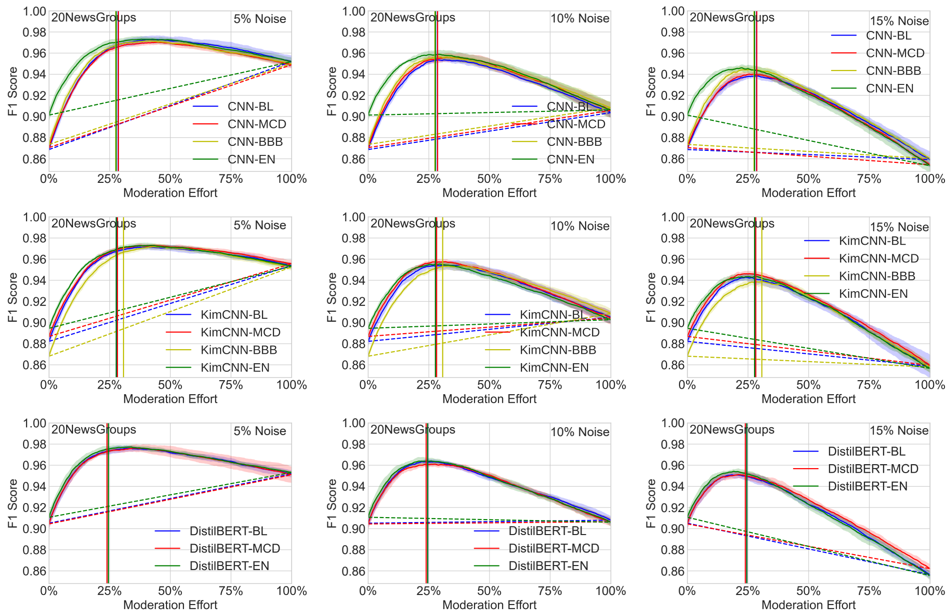


Figure A.2: Active Moderation under human noise (5%, 10%, and 15%) on the **20NewsGroups** dataset using various uncertainty modeling techniques. Vertical lines illustrate the saturation points.

Appendix B

Local Explanations

Label and Confidence	Text and Explanations	Type
Negative 0.51	An average thriller that keeps you guessing until the end, but falls short on execution.	$R_c(\epsilon_i)$
	An average thriller that keeps you guessing until the end, but falls short on execution.	$\phi(S_c(e_i))$
	An average thriller that keeps you guessing until the end, but falls short on execution.	$U(\epsilon_i)$
	An average thriller that keeps you guessing until the end, but falls short on execution.	$U(e_i)$
	An average thriller that keeps you guessing until the end, but falls short on execution.	$DUX(e_i)$
Negative 1.00	An absolute disaster of a film with weak writing, poor acting, and a predictable plot.	$R_c(\epsilon_i)$
	An absolute disaster of a film with weak writing, poor acting, and a predictable plot.	$\phi(S_c(e_i))$
	An absolute disaster of a film with weak writing, poor acting, and a predictable plot.	$U(\epsilon_i)$
	An absolute disaster of a film with weak writing, poor acting, and a predictable plot.	$U(e_i)$
	An absolute disaster of a film with weak writing, poor acting, and a predictable plot.	$DUX(e_i)$
Positive 0.90	A gripping tale of love, betrayal, and redemption that will leave you emotionally invested till the very end.	$R_c(\epsilon_i)$
	A gripping tale of love, betrayal, and redemption that will leave you emotionally invested till the very end.	$\phi(S_c(e_i))$
	A gripping tale of love, betrayal, and redemption that will leave you emotionally invested till the very end.	$U(\epsilon_i)$
	A gripping tale of love, betrayal, and redemption that will leave you emotionally invested till the very end.	$U(e_i)$
	A gripping tale of love, betrayal, and redemption that will leave you emotionally invested till the very end.	$DUX(e_i)$

Table B.1: Visualization of the different types of explanations obtained from BayLUXT. Each explanation consists of a heat-map visualizing different types of word attributions. A word at index i is highlighted according to the word relevance $R_c(\epsilon_i)$ the sequence relevance $\phi(S_c(e_i))$ the word uncertainty $U(\epsilon_i)$ and the sequence uncertainty $U(e_i)$, and DUX $DUX(e_i)$. Positive and negative sentiment words are highlighted in green and red respectively, while words that reduce and increase uncertainty are highlighted in blue and orange respectively.

Text 1 – Sentiment: *positive*, Probability: *68%*	Type
It was an excellent performance by the actors and a great setting. Unfortunately, the plot was terrible. I hope that the actors find new projects.	$R_c(\varepsilon_i)$
It was an excellent performance by the actors and a great setting. Unfortunately, the plot was terrible. I hope that the actors find new projects.	$DUX(e_i)$

Text 2 – Sentiment: *negative*, Probability: *64%*	Type
I am a big fan of Kevin Spacey, but this is a substandard film. If you think it looks quiet interesting or you have seen it and liked it go and check out 'John boorman is The General', it is basically about the same guy but is far superior in every way.	$R_c(\varepsilon_i)$
I am a big fan of Kevin Spacey, but this is a substandard film. If you think it looks quiet interesting or you have seen it and liked it go and check out 'John boorman is The General', it is basically about the same guy but is far superior in every way.	$DUX(e_i)$

Text 3 – Sentiment: *negative*, Probability: *60%*	Type
This movie is so bad it is almost worth watching. Trash at its very best. You really have to see it. It is so bad my sides hurt from laughing.	$R_c(\varepsilon_i)$
This movie is so bad it is almost worth watching. Trash at its very best. You really have to see it. It is so bad my sides hurt from laughing.	$DUX(e_i)$

Text 4 – Sentiment: *negative*, Probability: *97%*	Type
After 15 minutes of watching the movie I was asking myself what to do, leave the cinema, sleep or try to keep watching the movie to see if there was anything worth seeing. Finally, I watched the movie. What a waste of time, maybe I am not a 10 years old kid anymore.	$R_c(\varepsilon_i)$
After 15 minutes of watching the movie I was asking myself what to do, leave the cinema, sleep or try to keep watching the movie to see if there was anything worth seeing. Finally, I watched the movie. What a waste of time, maybe I am not a 10 years old kid anymore.	$DUX(e_i)$

Text 5 – Sentiment: *positive*, Probability: *91%*	Type
A gripping tale of love, betrayal, and redemption that will leave you emotionally invested till the very end.	$R_c(\varepsilon_i)$
A gripping tale of love, betrayal, and redemption that will leave you emotionally invested till the very end.	$DUX(e_i)$

Text 6 – Sentiment: *positive*, Probability: *57%*	Type
The opening is admittedly great. Drew Barrymore picking up the phone only to find herself talking to a stalker is inspired. What a shame about the rest of the film.	$R_c(\varepsilon_i)$
The opening is admittedly great. Drew Barrymore picking up the phone only to find herself talking to a stalker is inspired. What a shame about the rest of the film.	$DUX(e_i)$

Text 7 – Sentiment: *negative*, Probability: *60%*	Type
My girlfriend recommended this movie to me because it was the sequel to her favourite movie. She thought it was excellent. Sadly, I did not like it at all. It was the worst movie I have ever seen. What a shame.	$R_c(\varepsilon_i)$
My girlfriend recommended this movie to me because it was the sequel to her favourite movie. She thought it was excellent. Sadly, I did not like it at all. It was the worst movie I have ever seen. What a shame.	$DUX(e_i)$

Table B.2: Texts and explanations used within the human evaluation.

Appendix C

ChatGPT Prompts

C.1 Explaining Word Relevance

Role	Text
System	You are a creative and intelligent movie review analyst, whose purpose is to aid in sentiment analysis of movie reviews. A review will be provided to you, and you must classify the review as either 1 (positive) or 0 (negative), as well as your confidence in the score you chose. The confidence should be a decimal number between 0 and 1, with 0 being the lowest confidence and 1 being the highest confidence. Output this in the Python tuple format (<int classification>, <float confidence>). Then, analyze how important every single word and punctuation token in the review was to your classification. The importance should be a decimal number to three decimal places ranging from -1 to 1, with -1 implying a negative sentiment and 1 implying a positive sentiment. Provide a list of (<word or punctuation>, <float importance>) for each and every word and punctuation token in the sentence in a format of Python list of tuples. Each word or punctuation is separated by a space. It does not matter whether or not the sentence makes sense. Do your best given the sentence. The movie review will be encapsulated within <review> tags. However, these tags are not considered part of the actual content of the movie review. Example output: (<int classification>, <float confidence>) [(<word or punctuation>, <float importance>), (<word or punctuation>, <float importance>), ...]
User	<review> *It was an excellent performance by the actors and a great setting. Unfortunately, the plot was terrible. I hope that the actors find new projects.* <review>
System	(0, 0.85) [('It', 0.2), ('was', 0.1), ('an', 0.1), ('excellent', 0.9), ('performance', 0.8), ('by', 0.1), ('the', 0.1), ('actors', 0.9), ('and', 0.4), ('a', 0.1), ('great', 0.8), ('setting', 0.7), ('Unfortunately', -0.8), (',', 0.1), ('the', 0.1), ('plot', -0.9), ('was', -0.1), ('terrible', -0.9), ('.', 0.1), ('I', 0.1), ('hope', 0.6), ('that', 0.1), ('the', 0.1), ('actors', 0.8), ('find', 0.5), ('new', 0.3), ('projects', 0.4), ('.', 0.1)]

Table C.1: A prompt proposed by Huang et al. [163] to query a decoder-based LLM to extract feature attribution-like self-explanations for sentiment analysis. *Gray text in italics* is a placeholder for the input text. The blue text is the output of the LLM.

Role	Text
System	You are an expert in sentiment analysis of movie reviews. I give you a movie review and ask you to classify it as 1 (sentiment positive) or -1 (sentiment negative). Also provide your classification probability, which describes your confidence in your decision. Similar to the LIME local explanation technique, provide a sentiment score for each word, ranging from 1 (sentiment positive) to -1 (sentiment negative), which describes the relevance of that word to the overall sentiment of the review. A score of 0 would represent a neutral word. Please do not omit any words. The movie review will be encapsulated within <review> tags. Please format your response as the following array: (<sentiment>, <confidence>) [(<word or punctuation>, <float word-importance>,), (<word or punctuation>, <float word-importance>), ...]
User	<review> *It was an excellent performance by the actors and a great setting. Unfortunately, the plot was terrible. I hope that the actors find new projects.* <review>
System	(-1, 0.75) [('It', 0.0), ('was', 0.0), ('an', 0.0), ('excellent', 0.9), ('performance', 0.8), ('by', 0.0), ('the', 0.0), ('actors', 0.8), ('and', 0.0), ('a', 0.0), ('great', 0.7), ('setting', 0.6), ('Unfortunately', -0.8), (',', 0.0), ('the', 0.0), ('plot', -0.9), ('was', 0.0), ('terrible', -0.9), ('I', 0.0), ('hope', 0.7), ('that', 0.0), ('the', 0.0), ('actors', 0.8), ('find', 0.5), ('new', 0.0), ('projects', 0.6), ('.', 0.0)]

Table C.2: ChatGPT prompt for relevance explanation engineered by our self. *Gray text in italics* is a placeholder for the input text. The blue text is the output of the LLM.

C.2 Explaining Word Uncertainty

Role	Text
System	You are an expert in sentiment analysis of movie reviews. I give you a movie review and ask you to classify it as 1 (sentiment positive) or -1 (sentiment negative). Also provide your classification probability, which describes your confidence in your decision. Furthermore, provide the uncertainty score describing the classification uncertainty of the classification. Similar to a local explanation technique, provide an uncertainty score for each word, ranging from 1 (contributes certainty) to -1 (contributes uncertainty), which describes the contribution of that word's certainty/uncertainty to the overall prediction uncertainty of the review. A score of 0 would represent a neutral word. Please do not omit any words. The movie review will be encapsulated within <review> tags. Please format your response as the following array: (<sentiment>, <confidence>, <uncertainty>) [(<word or punctuation>, <float word-uncertainty>), (<word or punctuation>, <float word-uncertainty>), ...]
User	<review> *It was an excellent performance by the actors and a great setting. Unfortunately, the plot was terrible. I hope that the actors find new projects.* <review>
System	(-1, 0.75, 0.65) [('It', 0), ('was', 0), ('an', 0), ('excellent', 0.8), ('performance', 0.8), ('by', 0), ('the', 0), ('actors', 0.8), ('and', 0), ('a', 0), ('great', 0.8), ('setting', 0.8), ('Unfortunately', -0.8), (',', 0), ('the', 0), ('plot', -0.8), ('was', 0), ('terrible', -0.8), ('.', 0), ('I', 0), ('hope', 0), ('that', 0), ('the', 0), ('actors', 0), ('find', 0), ('new', 0), ('projects', 0), ('.', 0)]

Table C.3: ChatGPT prompt for uncertainty explanations. *Gray text in italics* is a placeholder for the input text. The blue text is the output of the LLM.

List of Figures

List of Tables

List of Publications

The thesis is based on the listed peer-reviewed publications. We have adopted parts of them verbatim:

- Andersen, J. S. and Maalej, W. "Design Patterns for Machine Learning-Based Systems With Humans in the Loop". In: *IEEE Software* (2024) 41(04), pp. 151–159.

- Andersen, J. S. and Zukunft, O. "Explaining Prediction Uncertainty in Text Classification: The DUX Approach". In: *Proceedings of the 7th International Conference on Natural Language Processing and Information Retrieval (NLPIR)*. ACM. 2023, pp. 57–62.

- Andersen, J. S. and Zukunft, O. "Towards Low-budget Real-time Active Learning for Text Classification via Proxy-based Data Selection". In: *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART)*. Vol. 3. SciTePress. 2023, pp. 25–33.

- Andersen, J. S. "Why Do We Need Domain-Experts for End-to-End Text Classication? An Overview". In: *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART)*. Vol. 3. SciTePress. 2023, pp. 17–24.

- Andersen, J. S. and Zukunft, O. "More Sustainable Text Classification via Uncertainty Sampling and a Human-in-the-Loop". In: *Lecture Notes in Computer Science (LNCS)*. Springer. 2022, pp. 201–225.

- Andersen, J. S. and Maalej, W. "Efficient, Uncertainty-based Moderation of Neural Networks Text Classifiers". In: *Findings of the Association for Computational Linguistics (ACL)*. 2022, pp. 1536–1546.

- Andersen, J. S. and Zukunft, O. "Towards More Reliable Text Classification on Edge Devices via a Human-in-the-Loop". In: *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART)*. Vol. 2. SciTePress. 2022, pp. 636–646.

- Andersen, J. S., Zukunft, O., and Maalej, W. “REM: Efficient Semi-Automated Real-Time Moderation of Online Forums”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations (ACL-IJCNLP)*. 2021, pp. 142–149

- Andersen, J. S., Schöner, T., and Maalej, W. “Word-Level Uncertainty Estimation for Black-Box Text Classifiers using RNNs”. In: *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*. 2020, pp. 5541–5546.

- Tropmann-Frick, M. and Andersen, J. S. “Towards Visual Data Science-An Exploration”. In: *Proceedings of the International Conference on Human Interaction and Emerging Technologies (IHIET)*. Springer. 2019, pp. 371–377.

In addition, my peer-reviewed publications which are not part of this thesis:

- Barbas, H., Soll, M., Andersen, J. S., Bender, E., Hamann, F., Haustermann, M., and Sitzmann, D. “The MINTFIT Computer Science Online Course”. In: *2022 IEEE German Education Conference (GeCon)*. IEEE. 2022, pp. 1–6.

- Haering, M., Andersen, J. S., Biemann, C., Loosen, W., Milde, B., Pietz, T., Stoecker, C., Wiedemann, G., Zukunft, O., and Maalej, W. “Forum 4.0: An Open-Source User Comment Analysis Framework”. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations (EACL)*. 2021, pp. 63–70.

- Andersen, J. S. “A User Centric Visual Analytics Framework for News Discussions”. In: *Proceedings of the Workshops of the EDBT/ICDT 2019 Joint Conference*. 2019.

- Andersen, J. S. and Zukunft, O. “Semi-clustering that Scales: An Empirical Evaluation of GraphX”. In: *2016 IEEE International Congress on Big Data (BigData Congress)*. IEEE. 2016, pp. 333–336.

- Andersen, J. S. and Zukunft, O. “Evaluating the Scaling of Graph-algorithms for Big Data using GraphX”. In: *2016 2nd International Conference on Open and Big Data (OBD)*. IEEE. 2016, pp. 1–8.

Bibliography

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P. A., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. "Tensorflow: A System for Large-scale Machine Learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. USENIX Association. 2016, pp. 265–283.

[2] Abdelrazek, A., Eid, Y., Gawish, E., Medhat, W., and Hassan, A. "Topic Modeling Algorithms and Applications: A Survey". In: *Information Systems* 112 (2023), p. 102131.

[3] Adnan, K. and Akbar, R. "An Analytical Study of Information Extraction From Unstructured and Multidimensional Big Data". In: *Journal of Big Data* 6.1 (2019), pp. 1–38.

[4] Agarwal, S., Godbole, S., Punjani, D., and Roy, S. "How Much Noise Is Too Much: A Study in Automatic Text Classification". In: *Seventh IEEE International Conference on Data Mining (ICDM)*. IEEE. 2007, pp. 3–12.

[5] Aitchison, J. *Language Change: Progress or Decay?* Cambridge University Press. 2001.

[6] Al-Jarrah, O. Y., Yoo, P. D., Muhaidat, S., Karagiannidis, G. K., and Taha, K. "Efficient Machine Learning for Big Data: A Review". In: *Big Data Research* 2.3 (2015), pp. 87–93.

[7] Al-Subaihin, A. A., Sarro, F., Black, S., Capra, L., and Harman, M. "App Store Effects on Software Engineering Practices". In: *IEEE Transactions on Software Engineering* 47.2 (2019), pp. 300–319.

[8] Amershi, S., Cakmak, M., Knox, W. B., and Kulesza, T. "Power to the People: The Role of Humans in Interactive Machine Learning". In: *AI Magazine* 35.4 (2014), pp. 105–120.

[9] Andersen, J. S. "Why Do We Need Domain-Experts for End-to-End Text Classification? An Overview." In: *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART)*. Vol. 3. SciTePress, 2023, pp. 17–24.

[10] Andersen, J. S. and Maalej, W. "Design Patterns for Machine Learning-Based Systems With Humans in the Loop". In: *IEEE Software* 41.04 (2024), pp. 151–159.

[11] Andersen, J. S. and Maalej, W. "Efficient, Uncertainty-based Moderation of Neural Networks Text Classifiers". In: *Findings of the Association for Computational Linguistics: ACL*. 2022, pp. 1536–1546.

[12] Andersen, J. S., Schöner, T., and Maalej, W. "Word-level Uncertainty Estimation for Black-box Text Classifiers Using RNNs". In: *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*. 2020, pp. 5541–5546.

[13] Andersen, J. S. and Zukunft, O. "Explaining Prediction Uncertainty in Text Classification: The DUX Approach". In: *Proceedings of the 2023 7th International Conference on Natural Language Processing and Information Retrieval (NLPIR)*. ACM. 2023, pp. 57–62.

[14] Andersen, J. S. and Zukunft, O. "More Sustainable Text Classification via Uncertainty Sampling and a Human-in-the-Loop". In: *Lecture Notes in Computer Science (LNCS)*. Springer. 2022, pp. 201–225.

[15] Andersen, J. S. and Zukunft, O. "Towards Low-Budget Real-Time Active Learning for Text Classification via Proxy-Based Data Selection." In: *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART)*. Vol. 3. SciTePress. 2023, pp. 25–33.

[16] Andersen, J. S. and Zukunft, O. "Towards More Reliable Text Classification on Edge Devices via a Human-in-the-Loop." In: *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART)*. Vol. 2. SciTePress. 2022, pp. 636–646.

[17] Andersen, J. S., Zukunft, O., and Maalej, W. "REM: Efficient Semi-automated Real-time Moderation of Online Forums". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*. 2021, pp. 142–149.

[18] Antoran, J., Bhatt, U., Adel, T., Weller, A., and Hernández-Lobato, J. M. "Getting a CLUE: A Method for Explaining Uncertainty Estimates". In: *International Conference on Learning Representations (ICLR)*. 2021.

[19] Arco, F. M. Plaza-del, Nozza, D., Hovy, D., et al. “Respectful or Toxic? Using Zero-shot Learning with Language Models to Detect Hate Speech”. In: *The 7th Workshop on Online Abuse and Harms (WOAH)*. 2023.

[20] Arnt, A. and Zilberstein, S. “Learning to Perform Moderation in Online Forums”. In: *Proceedings IEEE/WIC International Conference on Web Intelligence (WI)*. IEEE. 2003, pp. 637–641.

[21] Arora, S., Nyberg, E., and Rose, C. “Estimating Annotation Cost for Active Learning in a Multi-annotator Environment”. In: *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*. 2009, pp. 18–26.

[22] Arras, L., Horn, F., Montavon, G., Müller, K.-R., and Samek, W. “"What is Relevant in a Text Document?": An Interpretable Machine Learning Approach”. In: *PLOS ONE* 12.8 (2017), pp. 1–23.

[23] Arrieta, A. B., Rodríguez, N. D., Ser, J. D., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. “Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI”. In: *Information Fusion* 58 (2020), pp. 82–115.

[24] Ashmore, R., Calinescu, R., and Paterson, C. “Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges”. In: *ACM Computing Surveys (CSUR)* 54.5 (2021), pp. 1–39.

[25] Assimakopoulos, S., Muskat, R. V., Plas, L. van der, and Gatt, A. “Annotating for Hate Speech: The MaNeCo Corpus and Some Input from Critical Discourse Analysis”. In: *Proceedings of the 12th Language Resources and Evaluation Conference*. 2020, pp. 5088–5097.

[26] Attenberg, J. M., Ipeirotis, P. G., and Provost, F. “Beat the Machine: Challenging Workers to Find the Unknown Unknowns”. In: *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*. 2011.

[27] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. “On Pixel-wise Explanations for Non-linear Classifier Decisions by Layer-wise Relevance Propagation”. In: *PLOS ONE* 10.7 (2015), pp. 1–46.

[28] Bahdanau, D., Cho, K. H., and Bengio, Y. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *3rd International Conference on Learning Representations (ICLR)*. 2015.

[29] Bansal, G., Nushi, B., Kamar, E., Weld, D. S., Lasecki, W. S., and Horvitz, E. "Updates in Human-AI Teams: Understanding and Addressing the Performance/Compatibility Tradeoff". In: *Proceedings of the AAAI Conference on Artificial Intelligence.* Vol. 33. 01. 2019, pp. 2429–2437.

[30] Bansal, G., Wu, T., Zhou, J., Fok, R., Nushi, B., Kamar, E., Ribeiro, M. T., and Weld, D. "Does the Whole Exceed its Parts? The Effect of AI Explanations on Complementary Team Performance". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems.* 2021, pp. 1–16.

[31] Barandas, M., Folgado, D., Santos, R., Simão, R., and Gamboa, H. "Uncertainty-based Rejection in Machine Learning: Implications for Model Development and Interpretability". In: *Electronics* 11.3 (2022), p. 396.

[32] Bengio, Y., Simard, P., and Frasconi, P. "Learning Long-term Dependencies with Gradient Descent is Difficult". In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166.

[33] Bernard, J., Hutter, M., Zeppelzauer, M., Fellner, D., and Sedlmair, M. "Comparing Visual-interactive Labeling with Active Learning: An Experimental Study". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2017), pp. 298–308.

[34] Bernard, J., Zeppelzauer, M., Sedlmair, M., and Aigner, W. "VIAL: a Unified Process for Visual Interactive Labeling". In: *The Visual Computer* 34.9 (2018), pp. 1189–1207.

[35] Bhatt, U., Xiang, A., Sharma, S., Weller, A., Taly, A., Jia, Y., Ghosh, J., Puri, R., Moura, J. M., and Eckersley, P. "Explainable Machine Learning in Deployment". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency.* 2020, pp. 648–657.

[36] Bilgic, M. and Getoor, L. "Effective Label Acquisition for Collective Classification". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 2008, pp. 43–51.

[37] Binkhonain, M. and Zhao, L. "A Review of Machine Learning Algorithms for Identification and Classification of Non-functional Requirements". In: *Expert Systems with Applications: X* 1 (2019).

[38] Blanco-Justicia, A. and Domingo-Ferrer, J. "Machine Learning Explainability Through Comprehensible Decision Trees". In: *3rd International Cross-Domain Conference for Machine Learning and Knowledge Extraction (CD-MAKE).* Springer. 2019, pp. 15–26.

[39] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. "Weight Uncertainty in Neural Network". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1613–1622.

[40] Boberg, S., Schatto-Eckrodt, T., Frischlich, L., and Quandt, T. "The Moral Gatekeeper? Moderation and Deletion of User-generated Content in a Leading News Forum". In: *Media and Communication* 6.4 (2018), pp. 58–69.

[41] Boguslav, M. and Cohen, K. B. "Inter-Annotator Agreement and the Upper Limit on Machine Performance: Evidence from Biomedical Natural Language Processing." In: *Studies in Health Technology and Informatics* 245 (2017), pp. 298–302.

[42] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146.

[43] Bolstad, W. M. and Curran, J. M. *Introduction to Bayesian Statistics*. John Wiley & Sons. 2016.

[44] Borrellas, P. and Unceta, I. "The Challenges of Machine Learning and Their Economic Implications". In: *Entropy* 23.3 (2021), p. 275.

[45] Bouguelia, M.-R., Nowaczyk, S., Santosh, K., and Verikas, A. "Agreeing to Disagree: Active Learning with Noisy Labels without Crowdsourcing". In: *International Journal of Machine Learning and Cybernetics* 9 (2018), pp. 1307–1319.

[46] Boukhelifa, N., Bezerianos, A., and Lutton, E. "Evaluation of Interactive Machine Learning Systems". In: *Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent* (2018), pp. 341–360.

[47] Braiek, H. B. and Khomh, F. "Machine Learning Robustness: A Primer". In: *arXiv preprint arXiv:2404.00897* (2024).

[48] Breiman, L. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

[49] Brier, G. W. "Verification of Forecasts Expressed in Terms of Probability". In: *Monthly Weather Review* 78.1 (1950), pp. 1–3.

[50] Brodersen, K. H., Ong, C. S., Stephan, K. E., and Buhmann, J. M. "The Balanced Accuracy and its Posterior Distribution". In: *2010 20th International Conference on Pattern Recognition*. IEEE. 2010, pp. 3121–3124.

[51] Brodley, C. E. and Smyth, P. "Applying Classification Algorithms in Practice". In: *Statistics and Computing* 7.1 (1997), pp. 45–56.

[52] Brunk, J., Mattern, J., and Riehle, D. M. "Effect of Transparency and Trust on Acceptance of Automatic Online Comment Moderation Systems". In: *2019 IEEE 21st Conference on Business Informatics (CBI)*. Vol. 1. IEEE. 2019, pp. 429–435.

[53] Brutlag, J. D. and Meek, C. "Challenges of the Email Domain for Text Classification". In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*. Vol. 2000. 2000, pp. 103–110.

[54] Buckels, E. E., Trapnell, P. D., and Paulhus, D. L. "Trolls Just Want to Have Fun". In: *Personality and Individual Differences* 67 (2014), pp. 97–102.

[55] Buntine, W. L. and Weigend, A. S. "Bayesian Back-Propagation". In: *Complex Syst.* 5.6 (1991), pp. 603–643.

[56] Burkart, N. and Huber, M. F. "A Survey on the Explainability of Supervised Machine Learning". In: *Journal of Artificial Intelligence Research* 70 (2021), pp. 245–317.

[57] Burkhardt, S., Siekiera, J., and Kramer, S. "Semisupervised Bayesian Active Learning for Text Classification". In: *Bayesian Deep Learning Workshop at NeurIPS*. 2018.

[58] Card, M. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann. 1999.

[59] Castelvecchi, D. "Can we Open the Black Box of AI?" In: *Nature News* 538.7623 (2016), p. 20.

[60] Cecchini, M., Aytug, H., Koehler, G. J., and Pathak, P. "Making Words Work: Using Financial Text as a Predictor of Financial Events". In: *Decision Support Systems* 50.1 (2010), pp. 164–175.

[61] Chancellor, S. "Toward Practices for Human-centered Machine Learning". In: *Communications of the ACM* 66.3 (2023), pp. 78–85.

[62] Chapelle, O., Scholkopf, B., and Zien, A. "Semi-supervised Learning". In: *Cambridge, Massachusettes: The MIT Press View Article* 2 (2006).

[63] Chen, G. M. and Pain, P. "Normalizing Online Comments". In: *Journalism Practice* 11.7 (2017), pp. 876–892.

[64] Chen, J. and Ran, X. "Deep Learning With Edge Computing: A Review". In: *Proceedings of the IEEE* 107.8 (2019), pp. 1655–1674.

[65] Chen, J., Chen, Y., Du, X., Li, C., Lu, J., Zhao, S., and Zhou, X. "Big Data Challenge: A Data Management Perspective". In: *Frontiers of Computer Science* 7.2 (2013), pp. 157–164.

[66] Chen, X., Ragni, A., Liu, X., and Gales, M. J. "Investigating Bidirectional Recurrent Neural Network Language Models for Speech Recognition". In: *Proceedings of Interspeech 2017*. International Speech Communication Association (ISCA). 2017, pp. 269–273.

[67] Chen, Y.-R. and Chen, H.-H. "Opinion Spam Detection in Web Forum: A Real Case Study". In: *Proceedings of the 24th International Conference on World Wide Web*. 2015, pp. 173–183.

[68] Cheng, Y., Wang, D., Zhou, P., and Zhang, T. "Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges". In: *IEEE Signal Processing Magazine* 35.1 (2018), pp. 126–136.

[69] Choi, M., Park, C., Yang, S., Kim, Y., Choo, J., and Hong, S. R. "AILA: Attentive Interactive Labeling Assistant for Document Classification Through Attention-based Deep Neural Networks". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–12.

[70] Cohen, W. W. and Singer, Y. "Context-sensitive Learning Methods for Text Categorization". In: *ACM Transactions on Information Systems (TOIS)* 17.2 (1999), pp. 141–173.

[71] Coleman, C., Yeh, C., Mussmann, S., Mirzasoleiman, B., Bailis, P., Liang, P., Leskovec, J., and Zaharia, M. "Selection via Proxy: Efficient Data Selection for Deep Learning". In: *International Conference on Learning Representations (ICLR)*. 2020.

[72] Confalonieri, R., Coba, L., Wagner, B., and Besold, T. R. "A Historical Perspective of Explainable Artificial Intelligence". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 11.1 (2021), e1391.

[73] Corazza, M., Menini, S., Cabrio, E., Tonelli, S., and Villata, S. "A Multilingual Evaluation for Online Hate Speech Detection". In: *ACM Transactions on Internet Technology (TOIT)* 20.2 (2020), pp. 1–22.

[74] Cortes, C., DeSalvo, G., and Mohri, M. "Learning with Rejection". In: *International Conference on Algorithmic Learning Theory*. Springer. 2016, pp. 67–82.

[75] Cortes, C., Jackel, L. D., and Chiang, W.-P. "Limits on Learning Machine Accuracy Imposed by Data Quality". In: *Advances in Neural Information Processing Systems* 7 (1994).

[76] Culotta, A. and McCallum, A. "Reducing Labeling Effort for Structured Prediction Tasks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 5. 2005, pp. 746–751.

[77] Dai, S.-C., Xiong, A., and Ku, L.-W. "LLM-in-the-loop: Leveraging Large Language Model for Thematic Analysis". In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. 2023, pp. 9993–10001.

[78] D'Amour, A., Heller, K. A., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M. D., Hormozdiari, F., Houlsby, N., Hou, S., Jerfel, G., Karthikesalingam, A., Lucic, M., Ma, Y., McLean, C. Y., Mincu, D., Mitani, A., Montanari, A., Nado, Z., Natarajan, V., Nielson, C., Osborne, T. F., Raman, R., Ramasamy, K., Sayres, R., Schrouff, J., Seneviratne, M., Sequeira, S., Suresh, H., Veitch, V., Vladymyrov, M., Wang, X., Webster, K., Yadlowsky, S., Yun, T., Zhai, X., and Sculley, D. "Underspecification Presents Challenges for Credibility in Modern Machine Learning". In: *Journal of Machine Learning Research* 23 (2022), pp. 1–61.

[79] Danilevsky, M., Qian, K., Aharonov, R., Katsis, Y., Kawas, B., and Sen, P. "A Survey of the State of Explainable AI for Natural Language Processing". In: *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*. 2020, pp. 447–459.

[80] Davidson, T., Warmsley, D., Macy, M., and Weber, I. "Automated Hate Speech Detection and the Problem of Offensive Language". In: *Proceedings of the 11th International AAAI Conference on Web and Social Media*. ICWSM '17. 2017, pp. 512–515.

[81] Davis, J. and Goadrich, M. "The Relationship Between Precision-Recall and ROC Curves". In: *Proceedings of the 23rd International Conference on Machine Learning*. 2006, pp. 233–240.

[82] Day, D., Aberdeen, J., Hirschman, L., Kozierok, R., Robinson, P., and Vilain, M. "Mixed-initiative Development of Language Processing Systems". In: *Fifth Conference on Applied Natural Language Processing*. 1997, pp. 348–355.

[83] De, A., Okati, N., Zarezade, A., and Rodriguez, M. G. “Classification Under Human Assistance”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 7. 2021, pp. 5905–5913.

[84] Dellermann, D., Calma, A., Lipusch, N., Weber, T., Weigel, S., and Ebel, P. “The Future of Human-AI Collaboration: A Taxonomy of Design Knowledge for Hybrid Intelligence Systems”. In: *Proceedings of the 52nd Hawaii International Conference on System Sciences*. 2019.

[85] Demner-Fushman, D., Chapman, W. W., and McDonald, C. J. “What can Natural Language Processing do for Clinical Decision Support?” In: *Journal of Biomedical Informatics* 42.5 (2009), pp. 760–772.

[86] Deng, X., Li, Y., Weng, J., and Zhang, J. “Feature Selection for Text Classification: A Review”. In: *Multimedia Tools and Applications* 78.3 (2019), pp. 3797–3816.

[87] Der Kiureghian, A. and Ditlevsen, O. “Aleatory or Epistemic? Does it Matter?” In: *Structural Safety* 31.2 (2009), pp. 105–112.

[88] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Vol. 1. 2019, pp. 4171–4186.

[89] Diethe, T., Borchert, T., Thereska, E., Balle, B., and Lawrence, N. D. “Continual Learning in Practice”. In: *Continual Learning Workshop of 32nd Conference on Neural Information Processing Systems (NeurIPS)*. 2019.

[90] Diplaris, S., Papadopoulos, S., Kompatsiaris, I., Heise, N., Spangenberg, J., Newman, N., and Hacid, H. “"Making Sense of it All" An Attempt to Aid Journalists in Analysing and Filtering User Generated Content”. In: *Proceedings of the 21st International Conference on World Wide Web*. 2012, pp. 1241–1246.

[91] Dix, A., Finlay, J., Abowd, G. D., and Beale, R. *Human-computer Interaction*. Pearson Education. 2004.

[92] Djandji, M., Baly, F., Antoun, W., and Hajj, H. “Multi-task Learning using AraBert for Offensive Language Detection”. In: *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*. 2020, pp. 97–101.

[93] Doherty, R. A. and Sorenson, P. "Keeping Users in the Flow: Mapping System Responsiveness with User Experience". In: *Procedia Manufacturing* 3 (2015), pp. 4384–4391.

[94] Dor, L. E., Halfon, A., Gera, A., Shnarch, E., Dankin, L., Choshen, L., Danilevsky, M., Aharonov, R., Katz, Y., and Slonim, N. "Active Learning for BERT: An Empirical Study". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 7949–7962.

[95] Dorogovtsev, S. N. and Mendes, J. F. F. "Language as an Evolving Word Web". In: *Proceedings of the Royal Society of London. Series B: Biological Sciences* 268.1485 (2001), pp. 2603–2606.

[96] Dreiseitl, S. and Ohno-Machado, L. "Logistic Regression and Artificial Neural Network Classification Models: A Methodology Review". In: *Journal of Biomedical Informatics* 35.5-6 (2002), pp. 352–359.

[97] Du, M., Liu, N., Yang, F., Ji, S., and Hu, X. "On Attribution of Recurrent Neural Network Predictions via Additive Decomposition". In: *The World Wide Web Conference*. 2019, pp. 383–393.

[98] Dudley, J. J. and Kristensson, P. O. "A Review of User Interface Design for Interactive Machine Learning". In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 8.2 (2018), pp. 1–37.

[99] Dwivedi, Y. K., Ismagilova, E., Hughes, D. L., Carlson, J., Filieri, R., Jacobson, J., Jain, V., Karjaluoto, H., Kefi, H., Krishen, A. S., et al. "Setting the Future of Digital and Social Media Marketing Research: Perspectives and Research Propositions". In: *International Journal of Information Management* 59 (2021), p. 102168.

[100] Endert, A., Fiaux, P., and North, C. "Semantic Interaction for Visual Text Analytics". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2012, pp. 473–482.

[101] Endert, A., Hossain, M. S., Ramakrishnan, N., North, C., Fiaux, P., and Andrews, C. "The Human is the Loop: New Directions for Visual Analytics". In: *Journal of Intelligent Information Systems* 43.3 (2014), pp. 411–435.

[102] Espadoto, M., Martins, R. M., Kerren, A., Hirata, N. S., and Telea, A. C. "Toward a Quantitative Survey of Dimension Reduction Techniques". In: *IEEE Transactions on Visualization and Computer Graphics* 27.3 (2019), pp. 2153–2173.

[103] Fails, J. A. and Olsen Jr, D. R. "Interactive Machine Learning". In: *Proceedings of the 8th International Conference on Intelligent User Interfaces*. 2003, pp. 39–45.

[104] Fan, Q., Yu, Y., Yin, G., Wang, T., and Wang, H. "Where is the Road for Issue Reports Classification based on Text Mining?" In: *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE. 2017, pp. 121–130.

[105] Fang, F., Dutta, K., and Datta, A. "Domain Adaptation for Sentiment Classification in Light of Multiple Sources". In: *INFORMS Journal on Computing* 26.3 (2014), pp. 586–598.

[106] Fang, M., Zhao, H., Song, X., Wang, X., and Huang, S. "Using Bidirectional LSTM with BERT for Chinese Punctuation Prediction". In: *2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP)*. IEEE. 2019, pp. 1–5.

[107] Feng, D., Harakeh, A., Waslander, S. L., and Dietmayer, K. "A Review and Comparative Study on Probabilistic Object Detection in Autonomous Driving". In: *IEEE Transactions on Intelligent Transportation Systems* 23.8 (2021), pp. 9961–9980.

[108] Ferlitsch, A. *Deep Learning Patterns and Practices*. Simon and Schuster. 2021.

[109] Fredriksson, T., Mattos, D. I., Bosch, J., and Olsson, H. H. "Data Labeling: An Empirical Investigation into Industrial Challenges and Mitigation Strategies". In: *International Conference on Product-Focused Software Process Improvement*. Springer. 2020, pp. 202–216.

[110] Freeman, L. C. *Elementary Applied Statistics: For Students in Behavioral Science*. John Wiley & Sons. 1965.

[111] Frischlich, L., Boberg, S., and Quandt, T. "Comment Sections as Targets of Dark Participation? Journalists' Evaluation and Moderation of Deviant User Comments". In: *Journalism Studies* 20.14 (2019), pp. 2014–2033.

[112] Fucci, D., Mollaalizadehbahnemiri, A., and Maalej, W. "On Using Machine Learning to Identify Knowledge in API Reference Documentation". In: *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2019, pp. 109–119.

[113] Gal, Y. and Ghahramani, Z. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *International Conference on Machine Learning*. PMLR. 2016, pp. 1050–1059.

[114] Gal, Y., Islam, R., and Ghahramani, Z. "Deep Bayesian Active Learning with Image Data". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1183–1192.

[115] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. "A Survey on Concept Drift Adaptation". In: *ACM Computing Surveys (CSUR)* 46.4 (2014), pp. 1–37.

[116] Ganaie, M. A., Hu, M., Malik, A., Tanveer, M, and Suganthan, P. "Ensemble Deep Learning: A Review". In: *Engineering Applications of Artificial Intelligence* 115 (2022), p. 105151.

[117] Ganesh, A., Cao, J., Perkoff, E. M., Southwell, R., Palmer, M., and Kann, K. "Mind the Gap between the Application Track and the Real World". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. Vol. 2. 2023, pp. 1833–1842.

[118] Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., Shahzad, M., Yang, W., Bamler, R., and Zhu, X. X. "A Survey of Uncertainty in Deep Neural Networks". In: *Artificial Intelligence Review* (2023), pp. 1–77.

[119] Geifman, Y. and El-Yaniv, R. "Selective Classification for Deep Neural Networks". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 4885–4894.

[120] Ghesu, F. C., Georgescu, B., Mansoor, A., Yoo, Y., Gibson, E., Vishwanath, R. S., Balachandran, A., Balter, J. M., Cao, Y., Singh, R., Digumarthy, S. R., Kalra, M. K., Grbic, S., and Comaniciu, D. "Quantifying and Leveraging Predictive Uncertainty for Medical Image Assessment". In: *Medical Image Analysis* 68 (2021), p. 101855.

[121] Goan, E. and Fookes, C. "Bayesian Neural Networks: An Introduction and Survey". In: *Case Studies in Applied Bayesian Data Science* (2020), pp. 45–87.

[122] Gosiewska, A. and Biecek, P. "IBreakDown: Uncertainty of Model Explanations for Non-additive Predictive Models". In: *arXiv preprint arXiv :1903.11420* (2019).

[123] Gralinski, F., Wróblewska, A., Stanisławek, T., Grabowski, K., and Górecki, T. "GEval: Tool for Debugging NLP Datasets and Models". In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. 2019, pp. 254–262.

[124] Gray, M., Savelka, J., Oliver, W., and Ashley, K. "Can GPT Alleviate the Burden of Annotation?" In: *Legal Knowledge and Information Systems*. IOS Press, 2023, pp. 157–166.

[125] Grießhaber, D., Maucher, J., and Vu, N. T. "Fine-tuning BERT for Low-Resource Natural Language Understanding via Active Learning". In: *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*. 2020, pp. 1158–1171.

[126] Grimmelmann, J. "The Virtues of Moderation". In: *Yale Journal of Law and Technology* 17.1 (2015), p. 2.

[127] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., and Chen, T. "Recent Advances in Convolutional Neural Networks". In: *Pattern Recognition* 77 (2018), pp. 354–377.

[128] Gudivada, V., Apon, A., and Ding, J. "Data Quality Considerations for Big Data and Machine Learning: Going Beyond Data Cleaning and Transformations". In: *International Journal on Advances in Software* 10.1 (2017), pp. 1–20.

[129] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. "On Calibration of Modern Neural Networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1321–1330.

[130] Guo, G., Wang, H., Bell, D., Bi, Y., and Greer, K. "Using kNN Model for Automatic Text Categorization". In: *Soft Computing* 10.5 (2006), pp. 423–430.

[131] Guo, X., Yin, Y., Dong, C., Yang, G., and Zhou, G. "On the Class Imbalance Problem". In: *2008 Fourth International Conference on Natural Computation*. Vol. 4. IEEE. 2008, pp. 192–201.

[132] Guzman, E. and Maalej, W. "How do Users Like this Feature? A Fine Grained Sentiment Analysis of App Reviews". In: *2014 IEEE 22nd International Requirements Engineering Conference (RE)*. IEEE. 2014, pp. 153–162.

[133] Habibullah, K. M. and Horkoff, J. "Non-functional Requirements for Machine Learning: Understanding Current Use and Challenges in Industry". In: *2021 IEEE 29th International Requirements Engineering Conference (RE)*. IEEE. 2021, pp. 13–23.

[134] Haering, M., Andersen, J. S., Biemann, C., Loosen, W., Milde, B., Pietz, T., Stoecker, C., Wiedemann, G., Zukunft, O., and Maalej, W. "Forum 4.0: An Open-Source User Comment Analysis Framework". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. 2021, pp. 63–70.

[135] Han, J. and Moraga, C. "The Influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning". In: *International Workshop on Artificial Neural Networks*. Springer. 1995, pp. 195–201.

[136] Handelman, G. S., Kok, H. K., Chandra, R. V., Razavi, A. H., Huang, S., Brooks, M., Lee, M. J., and Asadi, H. "Peering Into the Black Box of Artificial Intelligence: Evaluation Metrics of Machine Learning Methods". In: *American Journal of Roentgenology* 212.1 (2019), pp. 38–43.

[137] Harbi, S. H. A., Tidjon, L. N., and Khomh, F. "Responsible Design Patterns for Machine Learning Pipelines". In: *arXiv preprint arXiv:2306.01788* (2023).

[138] Häring, M., Loosen, W., and Maalej, W. "Who is Addressed in this Comment? Automatically Classifying Meta-Comments in News Comments". In: *Proceedings of the ACM on Human-Computer Interaction* 2.CSCW (2018), pp. 1–20.

[139] Harman, M., Jia, Y., and Zhang, Y. "App Store Mining and Analysis: MSR for App Stores". In: *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE. 2012, pp. 108–111.

[140] Hawkins, D. M. *Identification of Outliers*. Vol. 11. Springer. 1980.

[141] Hayes, P. J. "Intelligent High-volume Text Processing using Shallow, Domain-specific Techniques". In: *Text-based Intelligent Systems: Current Research and Practice in Information Extraction and Retrieval* (1992), pp. 227–242.

[142] Haykin, S. *Neural Networks: A Comprehensive Foundation*. International Edition. Prentice Hall. 1999.

[143] He, J., Zhang, X., Lei, S., Chen, Z., Chen, F., Alhamadani, A., Xiao, B., and Lu, C. "Towards More Accurate Uncertainty Estimation In Text Classification". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 8362–8372.

[144] He, P., Liu, X., Gao, J., and Chen, W. "DeBERTa: Decoding-enhanced BERT with Disentangled Attention". In: *International Conference on Learning Representations (ICLR)*. 2021.

[145] He, X., Zhao, K., and Chu, X. "AutoML: A Survey of the State-of-the-Art". In: *Knowledge-Based Systems* 212 (2021), p. 106622.

[146] Heiland, L., Hauser, M., and Bogner, J. "Design Patterns for AI-based Systems: A Multivocal Literature Review and Pattern Repository". In: *arXiv preprint arXiv:2303.13173* (2023).

[147] Heimerl, F., Koch, S., Bosch, H., and Ertl, T. "Visual Classifier Training for Text Document Retrieval". In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (2012), pp. 2839–2848.

[148] Hendrycks, D. and Gimpel, K. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: *International Conference on Learning Representations (ICLR)* (2017).

[149] Hey, T., Keim, J., Koziolek, A., and Tichy, W. F. "Norbert: Transfer Learning for Requirements Classification". In: *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE. 2020, pp. 169–179.

[150] Hingmire, S., Chougule, S., Palshikar, G. K., and Chakraborti, S. "Document Classification by Topic Labeling". In: *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2013, pp. 877–880.

[151] Hinton, G. E. "Learning Distributed Representations of Concepts". In: *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*. Vol. 1. Amherst. 1986, p. 12.

[152] Hochreiter, S. and Schmidhuber, J. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (1997), pp. 1735–1780.

[153] Höferlin, B., Netzel, R., Höferlin, M., Weiskopf, D., and Heidemann, G. "Inter-active Learning of Ad-hoc Classifiers for Video Visual Analytics". In: *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE. 2012, pp. 23–32.

[154] Hoi, S. C., Jin, R., and Lyu, M. R. “Batch Mode Active Learning with Applications to Text Categorization and Image Retrieval”. In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1233–1248.

[155] Holzinger, A. “Interactive Machine Learning for Health Informatics: When do we Need the Human-in-the-Loop?” In: *Brain Informatics* 3.2 (2016), pp. 119–131.

[156] Honeycutt, D., Nourani, M., and Ragan, E. “Soliciting Human-in-the-Loop User Feedback for Interactive Machine Learning Reduces User Trust and Impressions of Model Accuracy”. In: *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*. Vol. 8. 2020, pp. 63–72.

[157] Hosseinzadeh, M., Wachal, A., Khamfroush, H., and Lucani, D. E. “Optimal Accuracy-time Trade-off for Deep Learning Services in Edge Computing Systems”. In: *ICC 2021-IEEE International Conference on Communications*. IEEE. 2021, pp. 1–6.

[158] Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. “Bayesian Active Learning for Classification and Preference Learning”. In: *arXiv preprint arXiv:1112.5745* (2011).

[159] Hu, Q., Guo, Y., Cordy, M., Xie, X., Ma, W., Papadakis, M., and Le Traon, Y. “Towards Exploring the Limitations of Active Learning: An Empirical Study”. In: *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE. 2021, pp. 917–929.

[160] Hu, R., Mac Namee, B., and Delany, S. J. “Active Learning for Text Classification with Reusability”. In: *Expert Systems with Applications* 45 (2016), pp. 438–449.

[161] Hu, S., Ding, N., Wang, H., Liu, Z., Wang, J., Li, J., Wu, W., and Sun, M. “Knowledgeable Prompt-tuning: Incorporating Knowledge into Prompt Verbalizer for Text Classification”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*. Vol. 1. 2022, pp. 2225–2240.

[162] Hu, Y. and Khan, L. “Uncertainty-aware Reliable Text Classification”. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 628–636.

[163] Huang, S., Mamidanna, S., Jangam, S., Zhou, Y., and Gilpin, L. H. “Can Large Language Models Explain Themselves? A Study of LLM-Generated Self-Explanations”. In: *arXiv preprint arXiv:2310.11207* (2023).

[164] Hüllermeier, E. and Waegeman, W. “Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods”. In: *Machine Learning* 110.3 (2021), pp. 457–506.

[165] Ikonomakis, M, Kotsiantis, S., and Tampakas, V. “Text Classification using Machine Learning Techniques.” In: *WSEAS Transactions on Computers* 4.8 (2005), pp. 966–974.

[166] ISO Central Secretary. *Software Engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Data Quality Model*. en. Standard ISO/IEC 25012:2008. International Organization for Standardization. 2018.

[167] Ittoo, A., Nguyen, L. M., and Bosch, A. van den. “Text Analytics in Industry”. In: *Computers in Industry* 78.C (2016), pp. 96–107.

[168] Janiesch, C., Zschech, P., and Heinrich, K. “Machine Learning and Deep Learning”. In: *Electronic Markets* 31.3 (2021), pp. 685–695.

[169] Jánoky, L. V., Levendovszky, J., and Ekler, P. “An Analysis on the Revoking Mechanisms for JSON Web Tokens”. In: *International Journal of Distributed Sensor Networks* 14.9 (2018).

[170] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., and Fung, P. “Survey of Hallucination in Natural Language Generation”. In: *ACM Computing Surveys* 55.12 (2023), pp. 1–38.

[171] Jiang, L., Liu, S., and Chen, C. “Recent Research Advances on Interactive Machine Learning”. In: *Journal of Visualization* 22 (2019), pp. 401–417.

[172] Joachims, T. “Text Categorization with Support Vector Machines: Learning with Many Relevant Features”. In: *European Conference on Machine Learning*. Springer. 1998, pp. 137–142.

[173] Jöhnk, J., Weißert, M., and Wyrtki, K. “Ready or Not, AI Comes—An Interview Study of Organizational AI Readiness Factors”. In: *Business & Information Systems Engineering* 63.1 (2021), pp. 5–20.

[174] Jospin, L. V., Laga, H., Boussaid, F., Buntine, W., and Bennamoun, M. “Hands-on Bayesian Neural Networks—A Tutorial for Deep Learning Users”. In: *IEEE Computational Intelligence Magazine* 17.2 (2022), pp. 29–48.

[175] Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. “Fasttext.zip: Compressing Text Classification Models”. In: *arXiv preprint arXiv:1612.03651* (2016).

[176] Joulin, A., Grave, É., Bojanowski, P., and Mikolov, T. "Bag of Tricks for Efficient Text Classification". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Vol. 2. 2017, pp. 427–431.

[177] Kadhim, A. I. "Survey on Supervised Machine Learning Techniques for Automatic Text Classification". In: *Artificial Intelligence Review* 52.1 (2019), pp. 273–292.

[178] Kallis, R., Di Sorbo, A., Canfora, G., and Panichella, S. "Predicting Issue Types on GitHub". In: *Science of Computer Programming* 205 (2021), p. 102598.

[179] Kallis, R., Di Sorbo, A., Canfora, G., and Panichella, S. "Ticket Tagger: Machine Learning Driven Issue Classification". In: *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. 2019, pp. 406–409.

[180] Kaplan, A. M. and Haenlein, M. "Users of the World, Unite! The Challenges and Opportunities of Social Media". In: *Business Horizons* 53.1 (2010), pp. 59–68.

[181] Karmaker, S. K., Hassan, M. M., Smith, M. J., Xu, L., Zhai, C., and Veeramachaneni, K. "AutoML to Date and Beyond: Challenges and Opportunities". In: *ACM Computing Surveys (CSUR)* 54.8 (2021), pp. 1–36.

[182] Karmakharm, T., Aletras, N., and Bontcheva, K. "Journalist-in-the-Loop: Continuous Learning as a Service for Rumour Analysis". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*. 2019, pp. 115–120.

[183] Kay, M., Patel, S. N., and Kientz, J. A. "How Good is 85%? A Survey Tool to Connect Classifier Evaluation to Acceptability of Accuracy". In: *Proceedings of the 33rd annual ACM Conference on Human Factors in Computing Systems*. 2015, pp. 347–356.

[184] Keim, D., Andrienko, G., Fekete, J.-D., Görg, C., Kohlhammer, J., and Melançon, G. "Visual Analytics: Definition, Process, and Challenges". In: *Information Visualization*. Springer, 2008, pp. 154–175.

[185] Keim, D., Kohlhammer, J., Ellis, G., and Mansmann, F. *Mastering the Information Age Solving Problems with Visual Analytics*. Eurographics Association. 2010.

[186] Keim, D. A. "Visual Exploration of Large Data Sets". In: *Communications of the ACM* 44.8 (2001), pp. 38–44.

[187] Keim, D. A., Mansmann, F., Oelke, D., and Ziegler, H. "Visual Analytics: Combining Automated Discovery with Interactive Visualizations". In: *11th International Conference of Discovery Science*. Springer. 2008, pp. 2–14.

[188] Kendall, A. and Gal, Y. "What Uncertainties do we Need in Bayesian Deep Learning for Computer Vision?" In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 5580–5590.

[189] Khalid, S., Khalil, T., and Nasreen, S. "A Survey of Feature Selection and Feature Extraction Techniques in Machine Learning". In: *2014 Science and Information Conference*. IEEE. 2014, pp. 372–378.

[190] Khan, A., Sohail, A., Zahoora, U., and Qureshi, A. S. "A Survey of the Recent Architectures of Deep Convolutional Neural Networks". In: *Artificial Intelligence Review* 53.8 (2020), pp. 5455–5516.

[191] Khomh, F., Adams, B., Cheng, J., Fokaefs, M., and Antoniol, G. "Software Engineering for Machine-Learning Applications: The Road Ahead". In: *IEEE Software* 35.5 (2018), pp. 81–84.

[192] Khurana, D., Koli, A., Khatter, K., and Singh, S. "Natural Language Processing: State of the Art, Current Trends and Challenges". In: *Multimedia Tools and Applications* (2022), pp. 1–32.

[193] Kieseberg, P., Schantl, J., Frühwirt, P., Weippl, E., and Holzinger, A. "Witnesses for the Doctor in the Loop". In: *Brain Informatics and Health: 8th International Conference BIH 2015*. Springer. 2015, pp. 369–378.

[194] Kim, B. and Pardo, B. "A Human-in-the-Loop System for Sound Event Detection and Annotation". In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 8.2 (2018), pp. 1–23.

[195] Kim, Y. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1746–1751.

[196] Kingma, D. P. and Ba, J. "Adam: A Method for Stochastic Optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[197] Kläs, M. and Vollmer, A. M. "Uncertainty in Machine Learning Applications: A Practice-driven Classification of Uncertainty". In: *Computer Safety, Reliability, and Security: SAFECOMP 2018 Workshops*. Vol. 37. Springer. 2018, pp. 431–438.

[198] Kolhatkar, V. and Taboada, M. "Constructive Language in News Comments". In: *Proceedings of the First Workshop on Abusive Language Online*. 2017, pp. 11–17.

[199] Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. "Supervised Machine Learning: A Review of Classification Techniques". In: *Emerging Artificial Intelligence Applications in Computer Engineering* 160.1 (2007), pp. 3–24.

[200] Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., and Brown, D. "Text Classification Algorithms: A Survey". In: *Information* 10.4 (2019), p. 150.

[201] Kreek, R. A. and Apostolova, E. "Training and Prediction Data Discrepancies: Challenges of Text Classification with Noisy, Historical Data". In: *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*. 2018, pp. 104–109.

[202] Kreps, J. *Questioning the Lambda Architecture*. Accessed May 5, 2024. 2014. URL: `https://www.oreilly.com/radar/questioning-the-lambda-architecture/`.

[203] Kreps, J., Narkhede, N., and Rao, J. "Kafka: A Distributed Messaging System for Log Processing". In: *Proceedings of the NetDB*. Vol. 11. 2011, pp. 1–7.

[204] Kullback, S. and Leibler, R. A. "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86.

[205] Kuo, C. C. J. and Madni, A. M. "Green Learning: Introduction, Examples and Outlook". In: *Journal of Visual Communication and Image Representation* 90 (2023), p. 103685.

[206] Kwak, Y. H. and Anbari, F. T. "Benefits, Obstacles, and Future of Six Sigma Approach". In: *Technovation* 26.5-6 (2006), pp. 708–715.

[207] Kwon, Y., Won, J.-H., Kim, B. J., and Paik, M. C. "Uncertainty Quantification using Bayesian Neural Networks in Classification: Application to Biomedical Image Segmentation". In: *Computational Statistics & Data Analysis* 142 (2020), p. 106816.

[208] Lai, S., Xu, L., Liu, K., and Zhao, J. "Recurrent Convolutional Neural Networks for Text Classification". In: *Twenty-ninth AAAI Conference on Artificial Intelligence*. 2015.

[209] Lakshmanan, V., Robinson, S., and Munn, M. *Machine Learning Design Patterns*. O'Reilly Media. 2020.

[210] Lakshminarayanan, B., Pritzel, A., and Blundell, C. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 6405–6416.

[211] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations". In: *International Conference on Learning Representations (ICLR)*. 2019.

[212] Lang, K. "Newsweeder: Learning to Filter Netnews". In: *Machine Learning Proceedings 1995*. Elsevier. 1995, pp. 331–339.

[213] LeCun, Y., Bengio, Y., and Hinton, G. "Deep Learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[214] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551.

[215] Lee, D. J.-L. and Macke, S. "A Human-in-the-loop Perspective on AutoML: Milestones and the Road Ahead". In: *IEEE Data Engineering Bulletin* (2020).

[216] Lee, K., Lee, K., Lee, H., and Shin, J. "Training Confidence-Calibrated Classifiers for Detecting Out-of-Distribution Samples". In: *International Conference on Learning Representations (ICLR)*. ICLR. 2018.

[217] Lemmens, J., Markov, I., and Daelemans, W. "Improving Hate Speech Type and Target Detection with Hateful Metaphor Features". In: *Proceedings of the Fourth Workshop on NLP for Internet Freedom: Censorship, Disinformation, and Propaganda*. 2021, pp. 7–16.

[218] Lertvittayakumjorn, P. and Toni, F. "Human-grounded Evaluations of Explanation Methods for Text Classification". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 5195–5205.

[219] Lewis, D. D. "Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval". In: *European Conference on Machine Learning*. Springer. 1998, pp. 4–15.

[220] Lewis, D. D. and Catlett, J. "Heterogeneous Uncertainty Sampling for Supervised Learning". In: *Machine Learning Proceedings 1994*. Elsevier. 1994, pp. 148–156.

[221] Lewis, D. D. and Gale, W. A. "A Sequential Algorithm for Training Text Classifiers". In: *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Springer. 1994, pp. 3–12.

[222] Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. "Rcv1: A new Benchmark Collection for Text Categorization Research". In: *Journal of Machine Learning Research* 5 (2004), pp. 361–397.

[223] Li, J., Chen, X., Hovy, E., and Jurafsky, D. "Visualizing and Understanding Neural Models in NLP". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016, pp. 681–691.

[224] Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., Yu, P. S., and He, L. "A Survey on Text Classification: From Traditional to Deep Learning". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 13.2 (2022), pp. 1–41.

[225] Li, X., Zhang, H., Ouyang, Y., Zhang, X., and Rong, W. "A Shallow BERT-CNN Model for Sentiment Analysis on Moocs Comments". In: *2019 IEEE International Conference on Engineering, Technology and Education (TALE)*. IEEE. 2019, pp. 1–6.

[226] Li, X. and Roth, D. "Learning Question Classifiers". In: *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*. ACL. 2002, 1–7.

[227] Li, Y. and Yang, T. "Word Embedding for Understanding Natural Language: A Survey". In: *Guide to Big Data Applications*. Springer, 2018, pp. 83–104.

[228] Lin, Z., Glass, L., Westover, M. B., Xiao, C., and Sun, J. "SCRIB: Set-classifier with Class-specific Risk Bounds for Blackbox Models". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 7. 2022, pp. 7497–7505.

[229] Link, D., Hellingrath, B., and Ling, J. "A Human-is-the-Loop Approach for Semi-Automated Content Moderation." In: *13th Proceedings of the International Conference on Information Systems for Crisis Response and Management*. 2016.

[230] Liu, R. and Gillies, D. F. "Overfitting in Linear Feature Extraction for Classification of High-dimensional Image Data". In: *Pattern Recognition* 53 (2016), pp. 73–86.

[231] Liu, X. "Intelligent Data Analysis: Issues and Challenges". In: *The Knowledge Engineering Review* 11.4 (1996), pp. 365–371.

[232] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[233] Liu, Z. and Heer, J. "The Effects of Interactive Latency on Exploratory Visual Analysis". In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 2122–2131.

[234] Loosen, W., Häring, M., Kurtanović, Z., Merten, L., Reimer, J., Roessel, L. van, and Maalej, W. "Making Sense of User Comments: Identifying Journalists' Requirements for a Comment Analysis Framework". In: *SCM Studies in Communication and Media* 6.4 (2018), pp. 333–364.

[235] Lowell, D., Lipton, Z. C., and Wallace, B. C. "Practical Obstacles to Deploying Active Learning". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 21–30.

[236] Lu, Q., Zhu, L., Xu, X., Whittle, J., Zowghi, D., and Jacquet, A. "Responsible AI Pattern Catalogue: A Collection of Best Practices for AI Governance and Engineering". In: *ACM Computing Surveys* (2023).

[237] Lüders, C. M., Pietz, T., and Maalej, W. "Automated Detection of Typed Links in Issue Trackers". In: *2022 IEEE 30th International Requirements Engineering Conference (RE)*. IEEE. 2022, pp. 26–38.

[238] Lundberg, S. M. and Lee, S.-I. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems* 30 (2017).

[239] Maalej, W., Kurtanović, Z., Nabil, H., and Stanik, C. "On the Automatic Classification of App Reviews". In: *Requirements Engineering* 21.3 (2016), pp. 311–331.

[240] Maalej, W. and Nabil, H. "Bug Report, Feature Request, or Simply Praise? On Automatically Classifying App Reviews". In: *2015 IEEE 23rd International Requirements Engineering Conference (RE)*. IEEE. 2015, pp. 116–125.

[241] Maalej, W., Nayebi, M., Johann, T., and Ruhe, G. "Toward Data-driven Requirements Engineering". In: *IEEE Software* 33.1 (2015), pp. 48–54.

[242] Maalej, W., Pham, Y. D., and Chazette, L. "Tailoring Requirements Engineering for Responsible AI". In: *Computer* 56.4 (2023), pp. 18–27.

[243] Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 2011, pp. 142–150.

[244] Maaten, L. Van der and Hinton, G. "Visualizing Data using t-SNE." In: *Journal of Machine Learning Research* 9.11 (2008).

[245] Malinin, A. and Gales, M. "Predictive Uncertainty Estimation via Prior Networks". In: *Advances in Neural Information Processing Systems* 31 (2018).

[246] Manosevitch, E. and Walker, D. "Reader Comments to Online Opinion Journalism: A Space of Public Deliberation". In: *International Symposium on Online Journalism*. Vol. 10. 2009, pp. 1–30.

[247] Martens, D. and Maalej, W. "Towards Understanding and Detecting Fake Reviews in App Stores". In: *Empirical Software Engineering* 24.6 (2019), pp. 3316–3355.

[248] Martin, G. L. and Corl, K. G. "System Response Time Effects on User Productivity". In: *Behaviour & Information Technology* 5.1 (1986), pp. 3–13.

[249] Medhat, W., Hassan, A., and Korashy, H. "Sentiment Analysis Algorithms and Applications: A Survey". In: *Ain Shams Engineering Journal* 5.4 (2014), pp. 1093–1113.

[250] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. "A Survey on Bias and Fairness in Machine Learning". In: *ACM Computing Surveys (CSUR)* 54.6 (2021), pp. 1–35.

[251] Mena, J., Pujol, O., and Vitrià, J. "A Survey on Uncertainty Estimation in Deep Learning Classification Systems from a Bayesian Perspective". In: *ACM Computing Surveys (CSUR)* 54.9 (2021), pp. 1–35.

[252] Mikolov, T., Chen, K., Corrado, G., and Dean, J. "Efficient Estimation of Word Representations in Vector Space". In: *International Conference on Learning Representations (ICLR)*. 2013.

[253] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. "Distributed Representations of Words and Phrases and Their Compositionality". In: *Advances in Neural Information Processing Systems* 26 (2013).

[254] Min, B., Ross, H., Sulem, E., Veyseh, A. P. B., Nguyen, T. H., Sainz, O., Agirre, E., Heintz, I., and Roth, D. "Recent Advances in Natural Language Processing via Large Pre-trained Language Models: A Survey". In: *ACM Computing Surveys (CSUR)* 56.2 (2023), pp. 1–40.

[255] Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., and Gao, J. "Deep Learning–based Text Classification: A Comprehensive Review". In: *ACM Computing Surveys (CSUR)* 54.3 (2021), pp. 1–40.

[256] Miok, K., Škrlj, B., Zaharie, D., and Robnik-Šikonja, M. "To BAN or not to BAN: Bayesian Attention Networks for Reliable Hate Speech Detection". In: *Cognitive Computation* (2022), pp. 1–19.

[257] Mitchell, T. M. *Machine Learning*. Vol. 1. 9. McGraw-hill. 1997.

[258] Monarch, R. M. *Human-in-the-Loop Machine Learning: Active Learning and Annotation for Human-centered AI*. Simon and Schuster. 2021.

[259] Montgomery, L., Lüders, C., and Maalej, W. "An Alternative Issue Tracking Dataset of Public Jira Repositories". In: *Proceedings of the 19th International Conference on Mining Software Repositories*. 2022, pp. 73–77.

[260] Moreno-Torres, J. G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N. V., and Herrera, F. "A Unifying View on Dataset Shift in Classification". In: *Pattern Recognition* 45.1 (2012), pp. 521–530.

[261] Morris, J., Lifland, E., Yoo, J. Y., Grigsby, J., Jin, D., and Qi, Y. "TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2020, pp. 119–126.

[262] Mosqueira-Rey, E., Hernández-Pereira, E., Alonso-Ríos, D., Bobes-Bascarán, J., and Fernández-Leal, Á. "Human-in-the-Loop Machine Learning: A State of the Art". In: *Artificial Intelligence Review* 56.4 (2023), pp. 3005–3054.

[263] Murarka, A., Radhakrishnan, B., and Ravichandran, S. "Classification of Mental Illnesses on Social Media using RoBERTa". In: *Proceedings of the 12th International Workshop on Health Text Mining and Information Analysis*. 2021, pp. 59–68.

[264] Nadkarni, P. M., Ohno-Machado, L., and Chapman, W. W. "Natural Language Processing: An Introduction". In: *Journal of the American Medical Informatics Association* 18.5 (2011), pp. 544–551.

[265] Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., and Muharemagic, E. "Deep Learning Applications and Challenges in Big Data Analytics". In: *Journal of Big Data* 2.1 (2015), pp. 1–21.

[266] Nakov, P., Ritter, A., Rosenthal, S., Sebastiani, F., and Stoyanov, V. "SemEval-2016 Task 4: Sentiment Analysis in Twitter". In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. 2016, pp. 1–18.

[267] Naseem, U., Razzak, I., and Eklund, P. W. "A Survey of Pre-processing Techniques to Improve Short-text Quality: A Case Study on Hate Speech Detection on Twitter". In: *Multimedia Tools and Applications* 80 (2021), pp. 35239–35266.

[268] Nauta, M., Trienes, J., Pathak, S., Nguyen, E., Peters, M., Schmitt, Y., Schlötterer, J., Keulen, M. van, and Seifert, C. "From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI". In: *ACM Computing Surveys* 55.13s (2023), pp. 1–42.

[269] Nayak, A. S., Kanive, A. P., Chandavekar, N., and Balasubramani, R. "Survey on Pre-processing Techniques for Text Mining". In: *International Journal of Engineering and Computer Science* 5.6 (2016), pp. 16875–16879.

[270] Neves, M. and Ševa, J. "An Extensive Review of Tools for Manual Annotation of Documents". In: *Briefings in Bioinformatics* 22.1 (2021), pp. 146–163.

[271] Neville, J., Jensen, D., Friedland, L., and Hay, M. "Learning Relational Probability Trees". In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2003, pp. 625–630.

[272] Nghiem, M.-Q., Baylis, P., and Ananiadou, S. "Paladin: An Annotation Tool Based on Active and Proactive Learning". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. 2021, pp. 238–243.

[273] Nguyen, A. T., Kharosekar, A., Krishnan, S., Krishnan, S., Tate, E., Wallace, B. C., and Lease, M. "Believe it or Not: Designing a Human-AI Partnership for Mixed-initiative Fact-checking". In: *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 2018, pp. 189–199.

[274] Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., and Chang, Y. "Abusive Language Detection in Online User Content". In: *Proceedings of the 25th international Conference on World Wide Web*. 2016, pp. 145–153.

[275] Nouh, M., Nurse, J. R., and Goldsmith, M. "Understanding the Radical Mind: Identifying Signals to Detect Extremist Content on Twitter". In: *2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE. 2019, pp. 98–103.

[276] Nowak, J., Taspinar, A., and Scherer, R. "LSTM Recurrent Neural Networks for Short Text and Sentiment Classification". In: *International Conference on Artificial Intelligence and Soft Computing*. Springer. 2017, pp. 553–562.

[277] Oliveira, A. S., Cecote, T. C., Silva, P. H., Gertrudes, J. C., Freitas, V. L., and Luz, E. J. "How Good Is ChatGPT For Detecting Hate Speech In Portuguese?" In: *Anais do XIV Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana*. SBC. 2023, pp. 94–103.

[278] Olston, C. and Mackinlay, J. D. "Visualizing Data with Bounded Uncertainty". In: *IEEE Symposium on Information Visualization (INFOVIS)*. IEEE. 2002, pp. 37–40.

[279] OpenAI. *ChatGPT-3.5*. Accessed May 5, 2024. URL: `https://chatgpt.com`.

[280] Otter, D. W., Medina, J. R., and Kalita, J. K. "A Survey of the Usages of Deep Learning for Natural Language Processing". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.2 (2020), pp. 604–624.

[281] Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., et al. "The PRISMA 2020 Statement: An Updated Guideline for Reporting Systematic Reviews". In: *BMJ* 372 (2021).

[282] Paleyes, A., Urma, R.-G., and Lawrence, N. D. "Challenges in Deploying Machine Learning: a Survey of Case Studies". In: *ACM Computing Surveys* 55.6 (2022), pp. 1–29.

[283] Park, D., Sachar, S., Diakopoulos, N., and Elmqvist, N. "Supporting Comment Moderators in Identifying High Quality Online News Comments". In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2016, pp. 1114–1125.

[284] Patterns Greensoftware Foundation. *Use energy efficient AI/ML models.* Accessed May 5, 2024. URL: `https://patterns.greensoftware.foundation/catalog/ai/energy-efficent-models`.

[285] Pavlopoulos, J., Malakasiotis, P., and Androutsopoulos, I. "Deep Learning for User Comment Moderation". In: *Proceedings of the First Workshop on Abusive Language Online*. 2017, pp. 25–35.

[286] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[287] Pennington, J., Socher, R., and Manning, C. D. "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543.

[288] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Vol. 1. 2018, pp. 2227–2237.

[289] Platt, J. "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods". In: *Advances in Large Margin Classifiers* 10.3 (1999), pp. 61–74.

[290] Porter, R., Theiler, J., and Hush, D. "Interactive Machine Learning in Data Exploitation". In: *Computing in Science & Engineering* 15.5 (2013), pp. 12–20.

[291] Poursabzi-Sangdeh, F., Boyd-Graber, J., Findlater, L., and Seppi, K. "Alto: Active Learning with Topic Overviews for Speeding Label Induction and Document Labeling". In: *Proceedings of the 54th Annual Meeting

of the Association for Computational Linguistics. Vol. 1. 2016, pp. 1158–1169.

[292] Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., Shyu, M.-L., Chen, S.-C., and Iyengar, S. S. “A Survey on Deep Learning: Algorithms, Techniques, and Applications”. In: *ACM Computing Surveys (CSUR)* 51.5 (2018), pp. 1–36.

[293] Prabhu, A., Dognin, C., and Singh, M. “Sampling Bias in Deep Active Classification: An Empirical Study”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 4058–4068.

[294] Preuveneers, D., Tsingenopoulos, I., and Joosen, W. “Resource Usage and Performance Trade-offs for Machine Learning Models In Smart Environments”. In: *Sensors* 20.4 (2020), p. 1176.

[295] Pronin, E. “Perception and Misperception of Bias in Human Judgment”. In: *Trends in Cognitive Sciences* 11.1 (2007), pp. 37–43.

[296] Qian, J., Bethke, A., Liu, Y., Belding, E., and Wang, W. Y. “A Benchmark Dataset for Learning to Intervene in Online Hate Speech”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 4757–4766.

[297] Quandt, T. “Dark Participation”. In: *Media and Communication* 6.4 (2018), pp. 36–48.

[298] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. “Language Models are Unsupervised Multitask Learners”. In: *OpenAI Blog* 1.8 (2019), p. 9.

[299] Rahm, E. and Do, H. H. “Data Cleaning: Problems and Current Approaches”. In: *IEEE Data Engineering Bulletin* 23.4 (2000), pp. 3–13.

[300] Rahman, M. S., Khomh, F., Hamidi, A., Cheng, J., Antoniol, G., and Washizaki, H. “Machine Learning Application Development: Practitioners’ Insights”. In: *Software Quality Journal* 31.4 (2023), pp. 1065–1119.

[301] Raj, C., Agarwal, A., Bharathy, G., Narayan, B., and Prasad, M. “Cyberbullying Detection: Hybrid Models Based on Machine Learning and Natural Language Processing Techniques”. In: *Electronics* 10.22 (2021), p. 2810.

[302] Rajput, G., Punn, N. S., Sonbhadra, S. K., and Agarwal, S. "Hate Speech Detection Using Static BERT Embeddings". In: *Proceedings of the 9th International Conference of Big Data Analytics*. 2021, pp. 67–77.

[303] Ramaswamy, H. G., Tewari, A., and Agarwal, S. "Consistent Algorithms for Multiclass Classification with an Abstain Option". In: *Electronic Journal of Statistics* 12.1 (2018), pp. 530–554.

[304] Ramponi, A. and Tonelli, S. "Features or Spurious Artifacts? Data-centric Baselines for Fair and Robust Hate Speech Detection". In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. ACL. 2022, pp. 3027–3040.

[305] Rattigan, M. J., Maier, M., and Jensen, D. "Exploiting Network Structure for Active Inference in Collective Classification". In: *Seventh IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2007, pp. 429–434.

[306] Ravitch, M. M. "Subjectivity in Decision Making: Common Problems and Limitations". In: *World Journal of Surgery* 13 (1989), pp. 281–286.

[307] Reimer, J., Häring, M., Loosen, W., Maalej, W., and Merten, L. "Content Analyses of User Comments in Journalism: A Systematic Literature Review Spanning Communication Studies and Computer Science". In: *Digital Journalism* 11.7 (2023), pp. 1328–1352.

[308] Reimers, N. and Gurevych, I. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 3982–3992.

[309] Ribeiro, M. T., Singh, S., and Guestrin, C. ""Why Should I Trust you?" Explaining the Predictions of any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 1135–1144.

[310] Riehle, D. M., Niemann, M., Brunk, J., Assenmacher, D., Trautmann, H., and Becker, J. "Building an Integrated Comment Moderation System–Towards a Semi-automatic Moderation Tool". In: *International Conference on Human-Computer Interaction*. Springer. 2020, pp. 71–86.

[311] Rizos, G., Papadopoulos, S., and Kompatsiaris, Y. "Predicting News Popularity by Mining Online Discussions". In: *Proceedings of the 25th International Conference Companion on World Wide Web*. 2016, pp. 737–742.

[312] Roelofs, R., Shankar, V., Recht, B., Fridovich-Keil, S., Hardt, M., Miller, J., and Schmidt, L. "A Meta-analysis of Overfitting in Machine Learning". In: *Advances in Neural Information Processing Systems* 32 (2019).

[313] Roetzel, P. G. "Information Overload in the Information Age: A Review of the Literature from Business Administration, Business Psychology, and Related Disciplines with a Bibliometric Approach and Framework Development". In: *Business Research* 12.2 (2019), pp. 479–522.

[314] Ross, B., Rist, M., Carbonell, G., Cabrera, B., Kurowsky, N., and Wojatzki, M. "Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis". In: *Proceedings of NLP4CMC III: 3rd Workshop on Natural Language Processing for Computer-Mediated Communication*. Ed. by M. Beißwenger, M. Wojatzki, and T. Zesch. 2016, pp. 6–9.

[315] Roy, S., Harshvardhan, A., Mukherjee, A., and Saha, P. "Probing LLMs for Hate Speech Detection: Strengths and Vulnerabilities". In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. 2023, pp. 6116–6128.

[316] Ruder, S. "An Overview of Gradient Descent Optimization Algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[317] Rudin, C. and Wagstaff, K. L. "Machine Learning for Science and Society". In: *Machine Learning* 95.1 (2014), pp. 1–9.

[318] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. "Learning Representations by Back-propagating Errors". In: *Nature* 323.6088 (1986), pp. 533–536.

[319] Sacha, D., Sedlmair, M., Zhang, L., Lee, J. A., Weiskopf, D., North, S., and Keim, D. "Human-centered Machine Learning Through Interactive Visualization". In: ESANN. 2016.

[320] Sacha, D., Senaratne, H., Kwon, B. C., Ellis, G., and Keim, D. A. "The Role of Uncertainty, Awareness, and Trust in Visual Analytics". In: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2015), pp. 240–249.

[321] Sacha, D., Stoffel, A., Stoffel, F., Kwon, B. C., Ellis, G., and Keim, D. A. "Knowledge Generation Model for Visual Analytics". In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 1604–1613.

[322] Salton, G., Wong, A., and Yang, C.-S. “A Vector Space Model for Automatic Indexing”. In: *Communications of the ACM* 18.11 (1975), pp. 613–620.

[323] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. “DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter”. In: *arXiv preprint arXiv:1910.01108* (2019).

[324] Satopaa, V., Albrecht, J., Irwin, D., and Raghavan, B. “Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior”. In: *2011 31st International Conference on Distributed Computing Systems Workshops*. IEEE. 2011, pp. 166–171.

[325] Scheffer, T., Decomain, C., and Wrobel, S. “Active Hidden Markov Models for Information Extraction”. In: *International Symposium on Intelligent Data Analysis*. Springer. 2001, pp. 309–318.

[326] Schick, T. and Schütze, H. “Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference”. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 2021, pp. 255–269.

[327] Schmidt, A. and Wiegand, M. “A Survey on Hate Speech Detection Using Natural Language Processing”. In: *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*. 2017, pp. 1–10.

[328] Seifert, C. and Granitzer, M. “User-based Active Learning”. In: *2010 IEEE International Conference on Data Mining Workshops*. IEEE. 2010, pp. 418–425.

[329] Seifert, C. and Lex, E. “A Novel Visualization Approach for Data-mining-related Classification”. In: *2009 13th International Conference Information Visualisation*. IEEE. 2009, pp. 490–495.

[330] Semberecki, P. and Maciejewski, H. “Deep Learning Methods for Subject Text Classification of Articles”. In: *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE. 2017, pp. 357–360.

[331] Sengupta, S., Chakraborti, T., Sreedharan, S., Vadlamudi, S. G., and Kambhampati, S. “Radar—A Proactive Decision Support System for Human-in-the-Loop Planning”. In: *2017 AAAI Fall Symposium Series*. 2017.

[332] Settles, B. "Active Learning Literature Survey". In: *Science* 10.3 (1995), pp. 237–304.

[333] Settles, B. "From Theories to Queries: Active Learning in Practice". In: *Active Learning and Experimental Design Workshop In Conjunction with AISTATS 2010*. PMLR. 2011, pp. 1–18.

[334] Shannon, C. E. "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423.

[335] Sheng, V. S., Provost, F., and Ipeirotis, P. G. "Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2008, pp. 614–622.

[336] Sheng, V. S. and Zhang, J. "Machine Learning with Crowdsourcing: A Brief Summary of the Past Research and Future Directions". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 1. 2019, pp. 9837–9843.

[337] Shneiderman, B. "The Eyes have it: A Task by Data Type Taxonomy for Information Visualizations". In: *The Craft of Information Visualization*. Elsevier, 2003, pp. 364–371.

[338] Shu, K., Sliva, A., Wang, S., Tang, J., and Liu, H. "Fake News Detection on Social Media: A Data Mining Perspective". In: *ACM SIGKDD Explorations Newsletter* 19.1 (2017), pp. 22–36.

[339] Siau, K. and Wang, W. "Building Trust in Artificial Intelligence, Machine Learning, and Robotics". In: *Cutter Business Technology Journal* 31.2 (2018), pp. 47–53.

[340] Siddhant, A. and Lipton, Z. C. "Deep Bayesian Active Learning for Natural Language Processing: Results of a Large-Scale Empirical Study". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 2904–2909.

[341] Sigurbergsson, G. I. and Derczynski, L. "Offensive Language and Hate Speech Detection for Danish". In: *Proceedings of the 12th Language Resources and Evaluation Conference*. 2020, pp. 3498–3508.

[342] Silva-Rodríguez, V., Nava-Muñoz, S. E., Castro, L. A., Martínez-Pérez, F. E., Pérez-González, H. G., and Torres-Reyes, F. "Classifying Design-level Requirements using Machine Learning for a Recommender of Interaction Design Patterns". In: *IET Software* 14.5 (2020), pp. 544–552.

[343] Slack, D., Hilgard, A., Singh, S., and Lakkaraju, H. "Reliable Post Hoc Explanations: Modeling Uncertainty in Explainability". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 9391–9404.

[344] Snow, R., O'connor, B., Jurafsky, D., and Ng, A. Y. "Cheap and Fast – But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks". In: *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2008, pp. 254–263.

[345] Sokolova, M. and Lapalme, G. "A Systematic Analysis of Performance Measures for Classification Tasks". In: *Information Processing & Management* 45.4 (2009), pp. 427–437.

[346] Sparck Jones, K. "A Statistical Interpretation of Term Specificity and its Application in Retrieval". In: *Journal of Documentation* 28.1 (1972), pp. 11–21.

[347] Sperrle, F., El-Assady, M., Guo, G., Borgo, R., Chau, D. H., Endert, A., and Keim, D. "A Survey of Human-Centered Evaluations in Human-Centered Machine Learning". In: *Computer Graphics Forum*. Vol. 40. 3. Wiley Online Library. 2021, pp. 543–568.

[348] Springer, N., Engelmann, I., and Pfaffinger, C. "User Comments: Motives and Inhibitors to Write and Read". In: *Information, Communication & Society* 18.7 (2015), pp. 798–815.

[349] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[350] Stanik, C., Haering, M., and Maalej, W. "Classifying Multilingual User Feedback using Traditional Machine Learning and Deep Learning". In: *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. IEEE. 2019, pp. 220–226.

[351] Stanik, C. and Maalej, W. "Requirements Intelligence with OpenReq Analytics". In: *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE. 2019, pp. 482–483.

[352] Stanik, C., Montgomery, L., Martens, D., Fucci, D., and Maalej, W. "A Simple NLP-based Approach to Support Onboarding and Retention in Open Source Communities". In: *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. 2018, pp. 172–182.

[353] Strubell, E., Ganesh, A., and McCallum, A. "Energy and Policy Considerations for Deep Learning in NLP". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 3645–3650.

[354] Sun, C., Qiu, X., Xu, Y., and Huang, X. "How to Fine-tune Bert for Text Classification?" In: *China National Conference on Chinese Computational Linguistics*. Springer. 2019, pp. 194–206.

[355] Tabassum, A. and Patil, R. R. "A Survey on Text Pre-processing & Feature Extraction Techniques in Natural Language Processing". In: *International Research Journal of Engineering and Technology (IRJET)* 7.06 (2020), pp. 4864–4867.

[356] Tambon, F., Laberge, G., An, L., Nikanjam, A., Mindom, P. S. N., Pequignot, Y., Khomh, F., Antoniol, G., Merlo, E., and Laviolette, F. "How to Certify Machine Learning Based Safety-critical Systems? A Systematic Literature Review". In: *Automated Software Engineering* 29.2 (2022), p. 38.

[357] Tang, X., Yang, K., Wang, H., Wu, J., Qin, Y., Yu, W., and Cao, D. "Prediction-uncertainty-aware Decision-making for Autonomous Vehicles". In: *IEEE Transactions on Intelligent Vehicles* 7.4 (2022), pp. 849–862.

[358] Tennant, G. *Six Sigma: SPC and TQM in Manufacturing and Services*. Routledge. 2017.

[359] Tian, Z., Cui, L., Liang, J., and Yu, S. "A Comprehensive Survey on Poisoning Attacks and Countermeasures in Machine Learning". In: *ACM Computing Surveys (CSUR)* 55.8 (2022), pp. 1–35.

[360] Titterington, D. M. "Bayesian Methods for Neural Networks and Related Models". In: *Statistical Science* (2004), pp. 128–139.

[361] Tolia, N., Andersen, D. G., and Satyanarayanan, M. "Quantifying Interactive User Experience on Thin Clients". In: *Computer* 39.3 (2006), pp. 46–52.

[362] Tomanek, K. and Morik, K. "Inspecting Sample Reusability for Active Learning". In: *Active Learning and Experimental Design Workshop In Conjunction with AISTATS 2010*. JMLR Workshop and Conference Proceedings. 2011, pp. 169–181.

[363] Tran, D., Liu, J. Z., Dusenberry, M. W., Phan, D., Collier, M., Ren, J., Han, K., Wang, Z., Mariet, Z. E., Hu, H., Band, N., Rudner, T. G. J., Nado, Z., Amersfoort, J. van, Kirsch, A., Jenatton, R., Thain, N., Buchanan, E. K., Murphy, K. P., Sculley, D., Gal, Y., Ghahramani, Z., Snoek, J., and Lakshminarayanan, B. "Plex: Towards Reliability using Pretrained Large Model Extensions". In: *First Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022*. 2022.

[364] Trivedi, S. K. and Dey, S. "An Enhanced Genetic Programming Approach for Detecting Unsolicited Emails". In: *2013 IEEE 16th International Conference on Computational Science and Engineering*. IEEE. 2013, pp. 1153–1160.

[365] Tropmann-Frick, M. and Andersen, J. S. "Towards Visual Data Science - An Exploration". In: *Human Interaction and Emerging Technologies: Proceedings of the 1st International Conference on Human Interaction and Emerging Technologies (IHIET)*. Springer. 2019, pp. 371–377.

[366] Tsiakas, K. and Murray-Rust, D. "Using Human-in-the-Loop and Explainable AI to Envisage new Future Work Practices". In: *Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments*. 2022, pp. 588–594.

[367] Van Der Sype, Y. S. and Maalej, W. "On Lawful Disclosure of Personal User Data: What Should App Developers do?" In: *2014 IEEE 7th International Workshop on Requirements Engineering and Law (RELAW)*. IEEE. 2014, pp. 25–34.

[368] Van Oordt, S. and Guzman, E. "On the Role of User Feedback in Software Evolution: a Practitioners' Perspective". In: *2021 IEEE 29th International Requirements Engineering Conference (RE)*. IEEE. 2021, pp. 221–232.

[369] Vapnik, V. "Principles of Risk Minimization for Learning Theory". In: *Advances in Neural Information Processing Systems* 4 (1991).

[370] Varshney, K. R. "Engineering Safety in Machine Learning". In: *2016 Information Theory and Applications Workshop (ITA)*. IEEE. 2016, pp. 1–5.

[371] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. "Attention is All you Need". In: *Advances in Neural Information Processing Systems* 30 (2017).

[372] Vazhentsev, A., Kuzmin, G., Shelmanov, A., Tsvigun, A., Tsymbalov, E., Fedyanin, K., Panov, M., Panchenko, A., Gusev, G., Burtsev, M., Avetisian, M., and Zhukov, L. “Uncertainty Estimation of Transformer Predictions for Misclassification Detection”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*. Vol. 1. 2022, pp. 8237–8252.

[373] Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. “Graph Attention Networks”. In: *6th International Conference on Learning Representations (ICLR)*. 2018.

[374] Venanzi, M., Guiver, J., Kazai, G., Kohli, P., and Shokouhi, M. “Community-based Bayesian Aggregation Models for Crowdsourcing”. In: *Proceedings of the 23rd International Conference on World Wide Web*. 2014, pp. 155–164.

[375] Vijayarani, S, Ilamathi, J, and Nithya, M. “Preprocessing Techniques for Text Mining-an Overview”. In: *International Journal of Computer Science & Communication Networks* 5.1 (2015), pp. 7–16.

[376] Vogelsang, A. and Borg, M. “Requirements Engineering for Machine Learning: Perspectives from Data Scientists”. In: *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. IEEE. 2019, pp. 245–251.

[377] Wanas, N., Magdy, A., and Ashour, H. “Using Automatic Keyword Extraction to Detect Off-topic Posts in Online Discussion Boards”. In: *Proceedings of Content Analysis for Web* 2 (2009).

[378] Wang, R. Y. and Strong, D. M. “Beyond Accuracy: What Data Quality Means to Data Consumers”. In: *Journal of Management Information Systems* 12.4 (1996), pp. 5–33.

[379] Wang, R., Li, Z., Cao, J., Chen, T., and Wang, L. “Convolutional Recurrent Neural Networks for Text Classification”. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2019, pp. 1–6.

[380] Wang, W., Zhang, M., Chen, G., Jagadish, H., Ooi, B. C., and Tan, K.-L. “Database Meets Deep Learning: Challenges and Opportunities”. In: *ACM SIGMOD Record* 45.2 (2016), pp. 17–22.

[381] Wang, Z. J., Choi, D., Xu, S., and Yang, D. “Putting Humans in the Natural Language Processing Loop: A Survey”. In: *Proceedings of the First Workshop on Bridging Human–Computer Interaction and Natural Language Processing*. 2021, pp. 47–52.

[382] Ware, M., Frank, E., Holmes, G., Hall, M., and Witten, I. H. "Interactive Machine Learning: Letting Users Build Classifiers". In: *International Journal of Human-Computer Studies* 55.3 (2001), pp. 281–292.

[383] Waseem, Z. "Are you a Racist or am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter". In: *Proceedings of the First Workshop on NLP and Computational Social Science*. 2016, pp. 138–142.

[384] Washizaki, H., Khomh, F., Guéhéneuc, Y.-G., Takeuchi, H., Natori, N., Doi, T., and Okuda, S. "Software-engineering Design Patterns for Machine Learning Applications". In: *Computer* 55.3 (2022), pp. 30–39.

[385] Washizaki, H., Uchida, H., Khomh, F., and Guéhéneuc, Y.-G. "Studying Software Engineering Patterns for Designing Machine Learning Systems". In: *2019 10th International Workshop on Empirical Software Engineering in Practice (IWESEP)*. IEEE. 2019, pp. 49–495.

[386] Watts, S., Shankaranarayanan, G., and Even, A. "Data Quality Assessment in Context: A Cognitive Perspective". In: *Decision Support Systems* 48.1 (2009), pp. 202–211.

[387] Whang, S. E. and Lee, J.-G. "Data Collection and Quality Challenges for Deep Learning". In: *Proceedings of the VLDB Endowment* 13.12 (2020), pp. 3429–3432.

[388] Wiechmann, M., Yimam, S. M., and Biemann, C. "ActiveAnno: General-purpose Document-level Annotation Tool with Active Learning Integration". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*. 2021, pp. 99–105.

[389] Wiethof, C. and Bittner, E. A. "Hybrid Intelligence–Combining the Human in the Loop with the Computer in the Loop: A Systematic Literature Review". In: *Proceedings of the 42nd International Conference on Information Systems (ICIS)*. Association for Information Systems. 2021.

[390] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., Platen, P. von, Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2020, pp. 38–45.

[391] Wondimu, N. A., Buche, C., and Visser, U. "Interactive Machine Learning: A State of the Art Review". In: *arXiv preprint arXiv:2207.06196* (2022).

[392] Wu, X., Xiao, L., Sun, Y., Zhang, J., Ma, T., and He, L. "A Survey of Human-in-the-Loop for Machine Learning". In: *Future Generation Computer Systems* (2022).

[393] Xiao, Y., Beschastnikh, I., Rosenblum, D. S., Sun, C., Elbaum, S., Lin, Y., and Dong, J. S. "Self-Checking Deep Neural Networks in Deployment". In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE. 2021, pp. 372–384.

[394] Xiong, H., Wu, J., and Liu, L. "Classification with Classoverlapping: A Systematic Study". In: *1st International Conference on E-Business Intelligence (ICEBI)*. Atlantis Press. 2010, pp. 303–309.

[395] Yan, Y., Chen, M., Shyu, M.-L., and Chen, S.-C. "Deep Learning for Imbalanced Multimedia Data Classification". In: *2015 IEEE International Symposium on Multimedia (ISM)*. IEEE. 2015, pp. 483–488.

[396] Yang, Y. and Liu, X. "A Re-examination of Text Categorization Methods". In: *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1999, pp. 42–49.

[397] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates. 2019.

[398] Ye, J., Gao, J., Li, Q., Xu, H., Feng, J., Wu, Z., Yu, T., and Kong, L. "ZeroGen: Efficient Zero-shot Learning via Dataset Generation". In: (2022), pp. 11653–11669.

[399] Yim, J., Joo, D., Bae, J., and Kim, J. "A Gift from Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4133–4141.

[400] Yu, Y., Si, X., Hu, C., and Zhang, J. "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures". In: *Neural Computation* 31.7 (2019), pp. 1235–1270.

[401] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. "Spark: Cluster Computing with Working Sets". In: *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud'10)*. Vol. 10. USENIX Association. 2010, pp. 95–105.

[402] Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., and Stoica, I. "Apache Spark: A Unified Engine for Big Data Processing". In: *Communications of the ACM* 59.11 (2016), pp. 56–65.

[403] Zanzotto, F. M. "Human-in-the-Loop Artificial Intelligence". In: *Journal of Artificial Intelligence Research* 64 (2019), pp. 243–252.

[404] Zhao, X., Huang, W., Huang, X., Robu, V., and Flynn, D. "BayLIME: Bayesian Local Interpretable Model-agnostic Explanations". In: *Uncertainty in Artificial Intelligence*. PMLR. 2021, pp. 887–896.

[405] Zhou, L., Pan, S., Wang, J., and Vasilakos, A. V. "Machine Learning on Big Data: Opportunities and Challenges". In: *Neurocomputing* 237 (2017), pp. 350–361.

[406] Zhu, J. and Ma, M. "Uncertainty-based Active Learning with Instability Estimation for Text Classification". In: *ACM Transactions on Speech and Language Processing (TSLP)* 8.4 (2012), pp. 1–21.

[407] Zhu, J., Wang, H., Yao, T., and Tsou, B. K. "Active Learning With Sampling by Uncertainty and Density for Word Sense Disambiguation and Text Classification". In: *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*. 2008, pp. 1137–1144.

Eidesstattliche Erklärung:

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Hamburg, den ....

Jakob Smedegaard Andersen