Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

FAKULTÄT
FÜR MATHEMATIK, INFORMATIK
UND NATURWISSENSCHAFTEN

Language Technology

DISSERTATION

Domain Defining Context: On Domain-Dependent Corpus Expansion and Contextualized Semantic Structuring

Steffen Remus

Language Technology
Department of Informatics
Faculty of Mathematics, Informatics and Natural Sciences

Universität Hamburg
Hamburg, Germany

A dissertation submitted for the degree of

Doctor rerum naturalium (Dr. rer. nat.)

April, 2023

Domain Defining Context: On Domain-Dependent Corpus Expansion and Contextualized Semantic Structuring

Dissertation submitted by: Steffen Remus

Date of Submission: April 24, 2023
Date of Disputation: June 19, 2023

Supervisor: Prof. Dr. Chris Biemann, Universität Hamburg

Committee:
1st Reviewer: Prof. Dr. Chris Biemann, Universität Hamburg
2nd Reviewer: Prof. Dr. Ricardo Usbeck, Universität Hamburg
Chair: Prof. Dr. Janick Edinger, Universität Hamburg

Universität Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics

Language Technology

Affidavit

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

I hereby declare upon oath that I have written the present dissertation independently and have not used further resources and aids than those stated in the dissertation.

April 24, 2023

Date

Signature

(Steffen Remus)

To my parents

Petra and Peter

Acknowledgements

I want to sincerely thank everybody that believed in me and this dissertation in times of progress and even more important, in times of struggle. This dissertation would not have been possible without the guidance, encouragement, and assistance of numerous individuals, and I am truly humbled by their contributions.

First and foremost, I would like to express my deepest appreciation to my supervisor, Chris. Your expertise, patience, and unwavering support have been invaluable to me. Your mentorship has shaped my research, challenged my ideas, and broadened my horizon. I am grateful for the countless hours you have dedicated to providing constructive feedback, guiding me through the ups and downs of the writing process, and encouraging me to push the limit and GET IT DONE.

I am forever grateful to Ido Dagan, Alfio Gliozzo, Abdel Labbi, and Animesh Mukherjee for giving me the opportunity to experience working with you from your institution. Working from abroad has always been a constant motivation and I cannot stress enough the appreciation I have for working under your guidance.

I am deeply indebted to my colleagues, collaborators, co-authors and fellow researchers, and 'smart*sses' ;) who have contributed to this dissertation in various ways. The many intellectual discussions during lunch, coffee breaks, feedback sessions, collaborations, and after-work activities have enriched my work and broadened my perspectives. I would like to extend my deepest thanks to Martin, your abiding support, understanding, and ability to lift my spirits have been an incredible gift. Whether it was our regular coffee breaks, concert or dart nights, or engaging conversations, you provided the perfect balance to the demands of my research. Thank you for your insightful input, stimulating discussions, and camaraderie throughout this research journey. Your expertise, faith, diverse viewpoints, and willingness to lend a helping hand have been instrumental in shaping the outcome of this work.

I would like to express my sincere appreciation to Gabriella for the valuable discussions and practical motivation during the final stages of this project. Your support and encouragement have been instrumental in pushing me toward the finish line. I am truly grateful for meeting you and the positive impact you have had on the completion of this endeavor. Thanks for lending an ear and your perfect understanding.

I am truly grateful to Josh for your friendship, constant motivation, and ability to bring a much-needed distraction to my life. Your presence has been a source of comfort and support during challenging times, and your infectious enthusiasm has kept me motivated throughout this journey. Thanks for keeping on bugging me.

I would also like to express my heartfelt thanks to Lucia. Your love, understanding, and unwavering belief in my abilities have sustained me throughout this demanding journey. Your presence in my life has been a constant source of

motivation and joy, and I am forever grateful for your support.

Lastly, I would like to express my deepest gratitude to my parents for their unwavering belief in me and their constant support in any life situation during the entire time of my academic endeavor. Your love, patience, and understanding have provided me with the strength and motivation to pursue this academic endeavor. Danke liebe Petra und lieber Peter für euch.

I am truly fortunate to have such a supportive network in my life. In conclusion, I am deeply grateful to all the individuals who have played a significant role in shaping this dissertation and my academic journey. Thanks to all of you and thanks to all of you who I forgot to mention here. I truly appreciate your belief in me in times of progress and even more important, in those times of struggle. Your support, guidance, friendship, and love have made a lasting impact on my life. Thank you for believing in me and for your invaluable contributions.

"It ain't what you say, it's the way that you say it, and the context in which you say it. Words are how you use them."

— Ludwig Wittgenstein (1889 – 1951)

Abstract

Natural language processing (NLP) is the task of processing potentially very large text collections and structuring the contained information so that it can be presented to the human user in a structured and condensed form. In linguistics and philosophy, it has been stated that communication between humans can be misleading and ambiguous and that the situational context is important to resolve the intended and expected meaning of utterances. As intelligent beings, however, humans can use the situational context to resolve ambiguities, such as body language, gestures, tone, surroundings, and so on. But using computers to achieve a goal based on natural language input or output is always prone to ambiguity on various levels because the situational context cannot be easily supplied. A computer task that involves NLP, e.g., to find documents given a query or classify documents into pre-defined classes such as spam or ham (not spam), often contains assumptions by the user, the human, that cannot be disambiguated easily by the computer without the provision of proper context.

In this dissertation, several studies have been conducted to incorporate context at various levels. First, we use the definition of a domain to set the context on a vocabulary basis, i.e., by using datasets that contain mainly content words from the desired domain, we narrow down the sample space of ambiguities. For instance, consider searching for the word 'virus' in a search engine: without any domain restriction, the best thing a computer can reply is the so-called major sense or mixed results based on the processed data, i.e., documents about organisms or computers or anything else will be returned if the underlying corpus was too generic. By restricting the underlying dataset to the biology domain, computers will reply mainly, or even exclusively in the ideal case, with documents from the biology domain. To collect domain corpora that can be used as a basis for NLP algorithms, we propose a simple yet effective technique for **focused web crawling** using statistical language models. Focused web crawling is resource-friendly since it ideally downloads only documents which are relevant to the domain of interest. The domain of interest is defined by a statistical language model created from a rather small corpus, e.g., we showed that a single Wikipedia article combined with a simple Kneser-Ney three-gram model is sufficient to guide the web-crawling process efficiently.

Another option to supply context is the computational representation of words, i.e., in NLP – which nowadays mainly relies on machine learning techniques – words, documents, or more general samples are represented by mathematical vectors, a.k.a. embeddings. Estimating such embeddings is an active area of research; in this dissertation, we retrofit so-called static word embeddings, i.e., the same static vector representation for each occurrence of a word, to so-called **sense embeddings**, i.e., an embedding in the vector space represents a single, distinct, sense of a word. Another popular representation is called **contextualized word**

embeddings, where the process of estimation is done by deep neural networks. Here, the so-called attention module usually provides a flow of information from one word of a sequence to another, i.e., when supplying a sequence, the word of interest is implicitly disambiguated, and the embedding mirrors the location shift based on the sequential context. We investigate the ability of contextualized word embeddings to model senses, analyze their performance in an **information retrieval** setup, and test their suitability to **induce semantic relations** from text. We show that contextualized embeddings are very suitable for modeling senses, retrieving similar sentences regarding a certain objective, and inducing semantic relations using unsupervised clustering methodologies. Finally, we can confirm that context certainly matters.

Zusammenfassung

Die Verarbeitung natürlicher Sprache (natural language processing, NLP) ist der Prozess potentiell sehr große Textmengen zu verarbeiten und die enthaltenen Informationen zu strukturieren, sodass sie für den menschlichen Benutzer in strukturierter und komprimierter Form dargestellt werden können. In der Linguistik und in der Philosophie wurde bereits festgestellt, dass die menschliche Kommunikation missverständlich und mehrdeutig sein kann, und dass der situative Kontext wichtig ist, um die beabsichtigte und erwartete Bedeutung von Äußerungen aufzulösen. Der Mensch, als intelligentes Wesen, ist, im Gegensatz zum Computer, in der Lage, unterschiedliche Kontexte zu nutzen um Mehrdeutigkeiten aufzulösen, so z. B. Körpersprache, Gesten, Tonfall, Umgebung, usw. Doch die Computerbenutzung ein Ziel zu erreichen, welches die Ein- oder Ausgabe von natürlicher Sprache beinhaltet, ist anfällig für Missverständnisse basierend auf Mehrdeutigkeiten auf verschiedenen Ebenen, da situativer Kontext nicht auf einfache Weise bereitgestellt werden kann. Eine computergestützte Aufgabe, die NLP voraussetzt, z. B. das Auffinden von Dokumenten anhand einer Suchanfrage, oder die Klassifizierung von Dokumenten in vordefinierte Klassen, enthält oft Annahmen des Benutzers, des Menschen, die von Computern ohne die Bereitstellung von Kontext nicht ohne weiteres disambiguiert werden können.

In dieser Arbeit wurden mehrere Studien durchgeführt, Kontext auf verschiedene Weisen einzubeziehen. Zunächst verwenden wir die Definition einer Domäne, um den Kontext auf der Grundlage des Vokabulars festzulegen, d. h. durch die Verwendung von Datensätzen, die hauptsächlich Inhaltswörter aus der gewünschten Domäne enthalten, wird der Ergebnisraum für Mehrdeutigkeiten eingeschränkt. Nehmen wir als Beispiel die Suche nach dem Wort ‘Virus’ in einer Suchmaschine: ohne jegliche Einschränkung der Domäne entspricht das Beste, was ein Computer antworten kann, dem so genannten Hauptsinn (major sense) bzw. es werden gemischte Ergebnisse auf der Grundlage der verarbeiteten Daten geliefert, d. h. es werden Dokumente über Organismen oder Computer oder irgendetwas anderem zurückgegeben, wenn der zugrundeliegende Korpus zu allgemein ist. Durch die Einschränkung des zugrundeliegenden Datensatzes auf Biologie-Dokumente z. B., werden Computer hauptsächlich, oder im Idealfall sogar ausschließlich, mit Dokumenten aus dem biologischen Bereich antworten. Um Domänenkorpora zu sammeln, die als Grundlage für NLP-Algorithmen verwendet werden können, schlagen wir eine einfache, aber effektive Technik für **fokussiertes Web-Crawling**, unter Verwendung statistischer Sprachmodelle, vor. Fokussiertes Web-Crawling ist ressourcenschonend, da es im Idealfall nur Dokumente herunterlädt, die für eine bestimmte Domäne relevant sind. Die Interessensdomäne wird durch ein statistisches Sprachmodell definiert, das aus einem relativ kleinen initialen Korpus erstellt werden kann. Wir haben z. B. gezeigt, dass ein einzelner Wikipedia-Artikel in Kombination mit einem einfachen Kneser-Ney-Drei-Gramm-Modell genügt,

um den fokussierten Web-Crawling-Prozess effizient zu steuern.

Eine weitere Möglichkeit, Kontext zu bereitzustellen, ist die computergestützte mathematische Repräsentation von Wörtern, d. h. im Bereich der natürlichen Sprachverarbeitung (NLP) – bei dem heutzutage hauptsächlich Techniken des maschinellen Lernens zum Einsatz kommen – werden Wörter, Dokumente oder allgemeinere Muster durch mathematische Vektoren, auch Einbettungen (embeddings) genannt, dargestellt. Das Lernen solcher Einbettungen ist ein aktives Forschungselement; in dieser Arbeit rüsten wir so genannte statische Einbettungen, bei denen dieselbe Vektorrepräsentation für jedes Vorkommen eines Wortes erstellt wird, zu sinnbehafteten Einbettungen, welche die verschiedenen Bedeutungen eines Wortes durch verschiedene Einbettungen im Vektorraum darstellt, auf. Eine weitere, heutzutage populärere, Repräsentation ist die kontextualisierte Worteinbettung, bei der das Lernen typischerweise von tiefen neuronalen Netzen übernommen wird. Hier sorgt üblicherweise das so genannte Attentionmodul für den Informationsfluss von einem Wort einer Sequenz zum Anderen, d. h. in einer gegebenen Sequenz werden Wörter implizit durch die Information der anderen Wörter disambiguiert, die Einbettung im Vektorraum spiegelt die Ortsverschiebung basierend auf dem sequenziellen Kontext wider. In dieser Arbeit analysieren wir die Fähigkeit kontextualisierter Einbettungen Bedeutungen zu modellieren. Weiterhin untersuchen wir in einem Information Retrieval Ansatz ihre Eignung semantische Relationen zu induzieren. Wir zeigen, dass kontextualisierte Einbettungen sehr gut geeignet sind Bedeutungen zu modellieren und semantisch ähnliche Sätze, mit Bezug einer bestimmten Klasse, hier semantische Relationen, durch unüberwachte Clustering-Methoden zu induzieren. Abschließend wird untermauert, dass Kontext eine große Rolle spielt.

Contents

List of Publications

Dissertation Related Publications

Dirk Goldhahn, **Steffen Remus**, Uwe Quasthoff, and Chris Biemann. 2014. Top-Level Domain Crawling for Producing Comprehensive Monolingual Corpora from the Web. In *Proceedings of the LREC-14 workshop on Challenges in the Management of Large Corpora (CMLC-2),* 10–14. Reykjavik, Iceland.

Omer Levy, **Steffen Remus**, Chris Biemann, and Ido Dagan. 2015. Do Supervised Distributional Methods Really Learn Lexical Inference Relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 970–976. Denver, CO, USA.

Varvara Logacheva, Denis Teslenko, Artem Shelmanov, **Steffen Remus**, Dmitry Ustalov, Andrey Kutuzov, Ekaterina Artemova, Chris Biemann, and Alexander Panchenko. 2020. Word sense disambiguation for 158 languages using word embeddings only. In *Proceedings of The 12th Language Resources and Evaluation Conference,* 5943–5952. Marseille, France.

Alexander Panchenko, Stefano Faralli, Eugen Ruppert, **Steffen Remus**, Hubert Naets, Cedrick Fairon, Simone P Ponzetto, and Chris Biemann. 2016. TAXI at SemEval-2016 Task 13: A Taxonomy Induction Method based on Lexico-Syntactic Patterns, Substrings and Focused Crawling. In *Proceedings of the 10th International Workshop on Semantic Evaluation,* 1320–1327. San Diego, CA, USA.

Steffen Remus. 2014. Unsupervised Relation Extraction of In-Domain Data from Focused Crawls. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics,* 11–20. Gothenburg, Sweden.

Steffen Remus and Chris Biemann. 2016. Domain-Specific Corpus Expansion with Focused Webcrawling. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016),* 23–28. Portorož, Slovenia.

———. 2018. Retrofitting Word Representations for Unsupervised Sense Aware Word Similarities. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018),* 1035–1041. Miyazaki, Japan.

Steffen Remus, Gerold Hintz, Darina Benikova, Thomas Arnold, Judith Eckle-Kohler, Christian M Meyer, Margot Mieskes, and Chris Biemann. 2016. EmpiriST: AIPHES Robust Tokenization and POS-Tagging for Different Genres. In *Proceedings of the 10th Web as Corpus Workshop (WAC-X),* 106–114. Berlin, Germany.

Steffen Remus, Gregor Wiedemann, Saba Anwar, Fynn Petersen-Frey, Seid Muhie Yimam, and Chris Biemann. 2022. More Like This: Semantic Retrieval with Linguistic Information. In *Proceedings of the 18th Conference on Natural Language Processing (KONVENS 2022),* 156–166. Potsdam, Germany.

Gregor Wiedemann, **Steffen Remus**, Avi Chawla, and Chris Biemann. 2019. Does BERT Make Any Sense? Interpretable Word Sense Disambiguation with Contextualized Embeddings. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019): Long Papers,* 161–170. Erlangen, Germany.

Other Publications

Rami Aly, **Steffen Remus**, and Chris Biemann. 2019. Hierarchical Multi-label Classification of Text with Capsule Networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop,* 323–330. Florence, Italy.

Chris Biemann, **Steffen Remus**, and Markus J Hofmann. 2015. Predicting word 'predictability' in cloze completion, electroencephalographic and eye movement data. In *Proceedings of the 12th International Workshop on Natural Language Processing and Cognitive Science,* 83–93. Krakow, Poland.

Jingyuan Feng, Özge Sevgili, **Steffen Remus**, Eugen Ruppert, and Chris Biemann. 2020. Supervised Pun Detection and Location with Feature Engineering and Logistic Regression. In *Proceedings of the 5th SwissText & 16th KONVENS Joint Conference 2020,* 3:1–6. Zurich, Switzerland.

Tim Fischer, **Steffen Remus**, and Chris Biemann. 2019. LT Expertfinder: An Evaluation Framework for Expert Finding Methods. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations),* 98–104. Minneapolis, MN, USA.

———. 2022. Measuring Faithfulness of Abstractive Summaries. In *Proceedings of the 18th Conference on Natural Language Processing (KONVENS 2022),* 63–73. Potsdam, Germany.

Markus J Hofmann, Chris Biemann, and **Steffen Remus**. 2016. Benchmarking n-grams, topic models and recurrent neural networks by cloze completions, EEGs and eye movements. Edited by Bernadette Sharp, Florence Sèdes, and Wiesław Lubaszewski. *Cognitive Approach to Natural Language Processing,* 197–215.

Markus J Hofmann, **Steffen Remus**, Chris Biemann, and Ralph Radach. 2019. Language models can outperform empirical predictability in predicting eye movement data. In *Proceedings of the 20th European Conference on Eye Movements (ECEM) 2019.* Alicante, Spain.

———. 2020. Language models explain word reading times better than empirical predictability. *PsyArXiv,* 1–77.

———. 2022. Language models explain word reading times better than empirical predictability. Edited by Massimo Stella. *Frontiers in Artificial Intelligence,* 1–20.

Dirk Johannßen, Chris Biemann, **Steffen Remus**, Timo Baumann, and David Scheffer. 2020. GermEval 2020 Task 1 on the Classification and Regression of Cognitive and Motivational style from Text. In *Proceedings of the GermEval 2020 Task 1 Workshop in conjunction with the 5th SwissText & 16th KONVENS Joint Conference 2020,* 1–10. Zurich, Switzerland (online).

Benjamin Milde, Tim Fischer, **Steffen Remus**, and Chris Biemann. 2021. MoM: Minutes of Meeting Bot. In *Proceedings of Interspeech 2021 Show&Tell,* 3311–3312. Brno, Czech Republic.

Seid Muhie Yimam, **Steffen Remus**, Alexander Panchenko, Andreas Holzinger, and Chris Biemann. 2017. Entity-Centric Information Access with the Human-in-the-Loop for the Biomedical Domains. In *Proceddings of the Biomedical NLP Workshop associated with RANLP 2017,* 42–48. Varna, Bulgaria.

Jinseok Nam, Christian Kirschner, Zheng Ma, Nicolai Erbs, Susanne Neumann, Daniela Oelke, **Steffen Remus**, Chris Biemann, Judith Eckle-Kohler, Johannes Fürnkranz, Iryna Gurevych, Marc Rittberger, and Karsten Weihe. 2014. Knowledge Discovery in Scientific Literature. In *Proceedings of the 12th Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2014),* 66–76. Hildesheim, Germany.

Steffen Remus, Rami Aly, and Chris Biemann. 2019. GermEval 2019 Task 1: Hierarchical Classification of Blurbs. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019),* 280–292. Erlangen, Germany.

Steffen Remus and Chris Biemann. 2013. Three Knowledge-Free Methods for Automatic Lexical Chain Extraction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 989–999. Atlanta, GA, USA.

Steffen Remus, Hanna Hedeland, Anne Ferger, Kristin Bührig, and Chris Biemann. 2018. EXMARaLDA meets WebAnno. In *Proceedings of the CLARIN Annual Conference 2018 (CAC 2018),* 1–5. Pisa, Italy.

———. 2019a. Annotation gesprochener Daten mit WebAnno-MM. In *Die 6. Jahrestagung des DHd e.V. 2019.* Frankfurt & Mainz, Germany.

———. 2019b. WebAnno-MM: EXMARaLDA meets WebAnno. In *Selected papers from the CLARIN Annual Conference 2018,* 166–172. Linköping Electronic Conference Proceedings 159.

Steffen Remus, Manuel Kaufmann, Kathrin Ballweg, Tatiana von Landesberger, and Chris Biemann. 2017. Storyfinder: Personalized Knowledge Base Construction and Management by Browsing the Web. In *CIKM '17: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management,* 2519–2522. Singapore, Singapore.

Gopalakrishnan Venkatesh, Abhik Jana, **Steffen Remus**, Özge Sevgili, Gopalakrishnan Srinivasaraghavan, and Chris Biemann. 2022. Using Distributional Thesaurus To Enhance Transformer-based Contextualized Representations for Low Resource Languages. In *Proceedings of the 37th ACM/SIGAPP Symposium On Applied Computing (ACM SAC), Special Track on Knowledge and Natural Language Processing (KNLP),* 845–852. online.

List of Figures

List of Tables

List of Listings

List of Abbreviations

ACC . . . . . Accuracy
AI . . . . . Artifical Intelligence
AMI . . . . . Adjusted Mutual Information
ARI . . . . . Adjusted Rand Index
BCE . . . . . Before Common Era
BERT . . . . . Bidirectional Encoder Representations from Transformers
BoW . . . . . bag-of-words
C . . . . . Completeness
cBOW . . . . . continuous bag-of-words
CC . . . . . Common Crawl
CNN . . . . . Convolutional Neural Network
COW . . . . . Corpora from the Web
CW . . . . . Chinese Whispers (Algorithm)
CWE . . . . . Contextualized Word Embedding
DDOS . . . . . Distributed Denial-Of-Service (attack)
DF . . . . . Document Frequency
DH . . . . . Distributional Hypothesis
DL . . . . . Deep Learning
DNN . . . . . Deep Neural Network
ELMo . . . . . Embeddings from Language Models
FMI . . . . . Fowlkes-Mallows Index
FN . . . . . False Negative
FP . . . . . False Positive
GloVe . . . . . Global Vectors
GPU . . . . . Graphical Processing Units
H . . . . . Homogeneity (cluster metric)
H . . . . . Entropy (statistics)
HAL . . . . . Hyperspace Analog to Memory
HITL . . . . . Human in the Loop
HTML . . . . . Hypertext Markup Language

IDF . . . . . Inverse Document Frequency
IR . . . . . Information Retrieval
JBT . . . . . JoBimText (German: Job im Text)
KB . . . . . Knowledge Base
KG . . . . . Knowlegde Graph
kNN . . . . . k-nearest neighbors
LDA . . . . . Latent Dirichlet Allocation
LISA . . . . . Lingustically-Informed Self-Attention
LLM . . . . . Large Language Model
LM . . . . . Language Model
LSA . . . . . Latent Semantic Analysis (a.k.a. LSI)
LSI . . . . . Latent Semantic Indexing (a.k.a. LSA)
LSTM . . . . . Long Short Term Memory
LU . . . . . Lexical Unit
mAP . . . . . Mean Average Precision
MCL . . . . . Markov Chain Clustering
MFS . . . . . Most Frequent Sense
ML . . . . . Machine Learning
MLE . . . . . maximum likelihood estimate
MLM . . . . . Masked Language Model
MM . . . . . MaxMax Clustering
MT . . . . . Machine Translation
MWE . . . . . Multi Word Expression
NELL . . . . . Never Ending Language Learner
NER . . . . . Named Entity Recognition
nf . . . . . non-focused
NLP . . . . . Natural Language Processing
NLU . . . . . Natural Language Understanding
NMI . . . . . Normalized Mutual Information
NN . . . . . Neural Network
NOD . . . . . Network of the Day
ODP . . . . . Open Directory Project: https://www.dmoz.org/
OOD . . . . . Out-of-Domain
OOV . . . . . out-of-vocabulary words
P@k . . . . . Precision at k
PCA . . . . . Principal Component Analysis

PLM . . . . . pre-trained language models
PLSA . . . . . Probabilistic Latent Semantic Analysis (a.k.a. PLSI)
PLSI . . . . . Probabilistic Latent Semantic Indexing (a.k.a. PLSA)
PoS . . . . . Part of Speech
PP . . . . . Perplexity
PPDB . . . . . Paraphrase Database
PPMI . . . . . Positive Pointwise Mutual Information
PU . . . . . Purity
PUF$_1$ . . . . . Purity F_1
RB . . . . . RoBERTa
RE . . . . . Relation Extraction.
RI . . . . . Rand Index
RNN . . . . . Recurrent Neural Network
RoBERTa . . . . Robustly optimized BERT approach
SBERT . . . . . Sentence BERT
SD . . . . . Structure Discovery
SGNS . . . . . Skip-Gram Negative Sampling
SOTA . . . . . state-of-the-art
SRL . . . . . Semantic Role Labeling
SVD . . . . . Singular Value Decomposition
SVM . . . . . Support Vector Machine
SWE . . . . . Static Word Embedding
sympat . . . . . Symmetric Patterns
synset . . . . . Set of Synonyms
t-SNE . . . . . T-distributed stochastic neighbor embedding
TAXI . . . . . The TAXonomy Induction system
TExEval . . . . . Shared task on Taxonomy Extraction Evaluation
TF . . . . . Term Frequency
TF-IDF . . . . . Term Frequency - Inverse Document Frequency
TL . . . . . Transfer Learning
tld . . . . . Top-Level Domain
TN . . . . . True Negative
TP . . . . . True Positive
UFSAC . . . . . Unification of Sense Annotated Corpora
URE . . . . . Unsupervised Relation Extraction
URL . . . . . Uniform Resource Locator

V . . . . . V-Measure
VSM . . . . . Vector Space Model
w2v Word2Vec
WaCKy . . . . . Web-as-Corpus Kool Yinitiative
WE . . . . . Word Embedding
WN . . . . . WordNet
WNGT . . . . . Princeton WordNet Annotated Gloss Tags
WSD . . . . . Word Sense Disambiguation
WSI . . . . . Word Sense Induction
WWW . . . . . World Wide Web

1 Introduction

Understanding natural human language is considered to be a key factor for current challenges to achieve artificial intelligence (AI; e.g., Nilsson, 2010). In his seminal work *"Computing Machinery and Intelligence"*, Turing (1950) proposed his infamous experimental setup where answers of a computer machine for a sequence of related or unrelated questions are tested if they are distinguishable from answers of intelligent humans. This test is commonly referred to as the *Turing Test*, and it is one of the most anticipated goals, if not the actual goal, of current AI research. To pass this test, a machine needs to process the information within the questions, identify its meaning and intention and reply with an answer that is syntactically reasonable as well as semantically coherent, even in the context of previous questions and answers. Natural Language Processing (NLP) can be considered as the branch of artificial intelligence (AI) that targets this kind of challenge, and Natural Language Understanding (NLU) is sometimes considered the sub-discipline that is mainly concerned with the meaning of texts, or – in some way – the semantics part of NLP (Allen, 1995). Here, semantics refers to the study of the relationship between words, phrases, or even longer expressions like sentences and their associated meanings.

In order for an autonomous computer system to pass the Turing test, we could say that various components have to conjoin, or better, the system must be able to handle multiple objectives jointly. Specific constraints apply for automatic systems as well as for humans to pass the Turing test: First of all, the questioner and the test subject must speak the same language; and second, the questioner and the test subject must have a common understanding of the domain they are communicating in. Misunderstandings are otherwise unavoidable.

In this work, we first address the importance of domain-aware mechanisms in NLP, particularly NLU. We borrow techniques from statistical language modeling to generate domain-dependent data from the World Wide Web by focusing web crawls on a particular domain of interest or topic. We show that domain-dependent downstream tasks (particularly unsupervised approaches for in-domain taxonomy induction) gain from domain-dependent data collected from the web (Chapter 2). Next, we retrofit static word embeddings to represent different senses and thus

create so-called static sense embeddings, i.e., words that genuinely have polysemic usages will be represented by multiple (distinct) vectors in the embedding space (Chapter 3). Also, in Chapter 3, we analyze the ability of contextual word embeddings from pre-trained language models to represent sense-related usages defined by the context they are currently used in. In Chapter 4, we expand those findings and hypothesize that contextual word embeddings can be used for the retrieval of semantic relations, where we also experiment with different vector representations using linguistic knowledge from automatic pre-processing steps. Converting the retrieval process into a nearest neighbor classification approach, we can show that only a few data samples are necessary to reach a decent and comparable performance. Chapter 5 continues this line of research and addresses the question of automatically clustering instances of verbs using their contextual vector representation – again exploiting linguistic knowledge for enhanced representational strength – by experimenting with a local-global word clustering approach.

The following sections will provide this dissertation's theoretical and historical foundations as well as background information.

1.1 Top-Down vs. Bottom-up

The concepts of *top-down* and *bottom-up* exist in many contexts; most important for this work is the AI perspective, which is inspired by *cognition*, and the *knowledge representation* perspective. In both cases, it explains the conceptual flow of information. It is either passed downwards from the top (possibly by a given entity) or assembled at the bottom and stacked upwards. In other terms, it can be understood as a progression from the most general kind of information (at the top) to the most specific one (at the bottom) or vice versa.

In cognition, top-down refers to perception based on one's prior knowledge or experience (schema-driven; Gregory, 1980), whereas bottom-up is understood as the interpretation of signals being determined solely by the stimuli experienced in the current (context-dependent) environment (data-driven; Gibson, 1971).

Top-down vs. bottom-up in AI dates back to Turing's posthumously published report "Intelligent Machinery" (1948), where Turing uses the cognitive top-down and bottom-up processes to propose a neuron-inspired, intelligent computing machine. Similar to the cognitive science definition, top-down refers to *schema-driven* or the *symbolic approach*, and bottom-up refers to a *data-driven* or *connectionist* approach. In the symbolic approach, rules and symbols are defined manually to decompose a complex problem into smaller, easier-to-solve problems; in computer science, this is also known as *divide-and-conquer*. On the other hand, the connectionist approach is defined by abstract (hidden) units connected by learning directly from the data.

In knowledge representation, the top refers to where the most general item is located, and the bottom refers to where the most specific items, e.g., the unprocessed data, are located. We adopt definitions by Sowa (2000) and Shapiro (1987), where top-down and bottom-up refers to the construction or the design of *ontologies*. Here, top-down refers to the process of building so-called *heavy-weight ontologies* that are manually crafted and thus very precise but incomplete by design

since they try to model the entire real-world knowledge. The first step in the conceptualization is often the design of a certain schema which is then applied to the concepts being modeled. Bottom-up refers to building so-called *light-weight ontologies*, which are usually also domain-ontologies and designed with a specific task in mind. Thus, concepts are often represented by the underlying data that is to be represented by the data-driven approach, e.g., words. The hierarchy of abstract concepts is usually relatively shallow, and the design is primarily schema-less. Automatic approaches, such as proposed by Biemann (2012), can support the creation process. The current work closely follows Biemann's (2012) bottom-up *structure discovery* (SD) approaches and turns to bring structure to loose words in an unsupervised fashion.

1.2 Historical Foundations in Linguistics

Linguistics has a history as a scientific discipline for a couple of centuries, which can still be considered a relatively young age within the humanities academic disciplines. *Computational linguistics*, where computers are used to aid in the analysis of written and spoken language, has a much shorter timeline, being an active discipline for a little more than half a century. But it has only been for a little more than two decades that *language understanding* reaches an active industry awareness and touches the lives of billions of people, even if they are unaware of its immediate and indispensable significance. It has only been during the last decade or so that *language technology* is evolving to a level that can even surpass human performance[1]. Due to those reasons, researchers in AI, particularly also NLP, are confronted with a new responsibility for society. Still, the immersive success of strong and robust industrial applications is based on practical and theoretical progress with core foundations also in knowledge representation, which is rooted in the study of meaning in philosophy, dating back several millennia. It has also influenced new scientific areas, e.g., the field of cognitive science, an interdisciplinary scientific area loosely affected by linguistics, psychology, philosophy, neuroscience, anthropology, artificial intelligence and probably even more.

Early works in linguistics often had opposing views, and different camps were literally fighting for recognition of their angle. The following paragraphs will give a broad overview with the most relevant, but not continuous or cumulative, milestones and pertinent achievements for this work as mainly summarized by Campbell (2017).

Grammar - The Beginning of Linguistics The first linguistically recognized texts were composed around 1900 BCE by the first notably accepted civilizations: the *Sumerians* and the *Akkadians*. The Sumerian language, which then shifted to Akkadian, lasted for roughly 2,500 years, and the discovered texts were mainly of religious nature, recipes, or legal texts, but they still provided a sound basis for further studies. Grammatical analysis of those texts then evolved around 600-500 BCE, which involved the origination of grammatical paradigms and multi-

1. ...for some tasks.

lingual dictionaries. In parallel, descriptions of Sanskrit also date back to 500 BCE, and during that time, Grammar was considered the most prestigious science in India, which resulted in the Hindu grammatical tradition. Greek grammatical tradition originates in the works of Homer (ca. 850 BCE), and themes found herein have also been found in works from Plato, Aristotle, and the Stoics. The most important ideas, such as morphology (word structure), etymology (word origin), parts-of-speech (structural sentence components of words), grammatical categories (structural elements of sentences across words), the relation between language and thought (semantics), etc. have persisted throughout the history and are still somewhat valid. E.g., the parts-of-speech (POS) categories have varied only slightly, and this is probably due to the adaptation to other languages. Only recently, within the last decade, several efforts have been made to build a unified representation, which can be used to map POS tags of one language to tags of another language, namely the *universal dependencies project*[2]. The main reason those efforts have been undertaken recently is probably of a technical nature, because of the rise of computational models for e.g., machine translation (MT).

Greek philosophers studied aspects of language mainly in the context of general knowledge, i.e., the *philosophia*. The term *linguistics* was first coined by the *Stoics* around 300 BCE, where initially the following three areas of interest were formed:

1. *Etymology*: the study of the origin of words
2. *Phonetics*: the study of characters and their relation to sound utterances
3. *Grammar*: the study of linguistic categories and terminology

The Stoics mainly studied the question if there is a "natural" relation between a word and its meaning or if this relationship is, in fact, a matter of convention, and to what extent Greek is characterized by such regularity, resp. irregularity. Two controversies have then emerged:

1. *Physis* (nature, Socrates, Stoics) vs. *nomos* (convention, Aristotles): the question was whether the meaning of a word originated in nature, i.e., words are formed by imitating things that exist, or whether convention, i.e., the use of a word (within certain situational contexts) contributes to a common understanding and give it its final meaning.
2. *Analogy* (Aristotles) vs. *Anomaly* (Stoics): the debate evolved around language acquisition, i.e., are new concepts learned by comparing them by (relational) similarity with other, known concepts (analogy), or are they learned by finding differences (differentiating) to other known concepts (anomaly).

Because of those debates, more detailed analyses of *form* and *meaning* have been conducted, which could be seen as the actual origin of *linguistics* as a field of study, and this then formed another controversy of two models of scientific explanation:

1. *Empiricist view*: accurate records are collected, and data analysis is the basis of hypotheses and explanations; the methodology is *inductive*.

2. https://universaldependencies.org/

2. *Epistemic view*: science is treated as a cognitive activity, where theories are built based on passed-down information, and new knowledge is acquired by the cognitive process of the individual scientist, the methodology is rather *deductive*.

Perspective

This work is located in:

nomos because we apply contextual learning methodologies, i.e., the meaning of a word is represented by the conventional use of the word and its surrounding context, and not dictated by an oracle;

analogy because throughout this work, we represent the sense of a word by a collection of other words which share a common concept;

empiricist view because we apply a data-driven, inductive bottom-up approach instead of a deductive top-down approach.

Modern Linguistics (Structuralism) Although research in the linguistic field and the evolution of languages continued throughout history, no particularly new ideas have been formed that have not been addressed already by the Greeks, with minor milestones in *a*) the middle ages, e.g., the use of Latin as *Lingua Franca* for educational and intellectual discourse, and *b*) the Renaissance period, e.g., by translating religious documents to a variety of commonly used languages or the discovery of new continents and thus the exploration of previously unknown languages.

A breakthrough has then taken place with Ferdinand de Saussure's seminal work *"Cours de linguistique générale"* (1983; orig. 1916). De Saussure is widely considered the founder of *modern linguistics* or, more precisely, *structuralism*. Among many thorough discussions regarding new approaches appearing by the end of the 19th century, which can be found in his lecture notes published by his disciples in 1916 after his death, he favored and promoted the idea of *langue* vs. *parole*. i.e., linguistics should be primarily concerned with the study of systematic regularities of the abstract language system shared by members of a language community (langue) rather than the concrete use of language by an individual (parole; Kortmann, 2005). At the heart of the *structuralism* idea stands the study and description of individual elements of the abstract language system, as well as the relationships between those elements on all structural levels, such as sounds, words, sentences, constituents, etc.

De Saussure distinguishes two basic types of relationships that exist in any language-related system:

1. *syntagmatic* relations: relationships of "chain", or combination,

2. *paradigmatic* relations: relationships of "choice", or interchangeability

In the sound system, for example, a paradigmatic relation exists between the initial sounds of, e.g., *ban, can, fan, van, ...* and a syntagmatic relation holds between

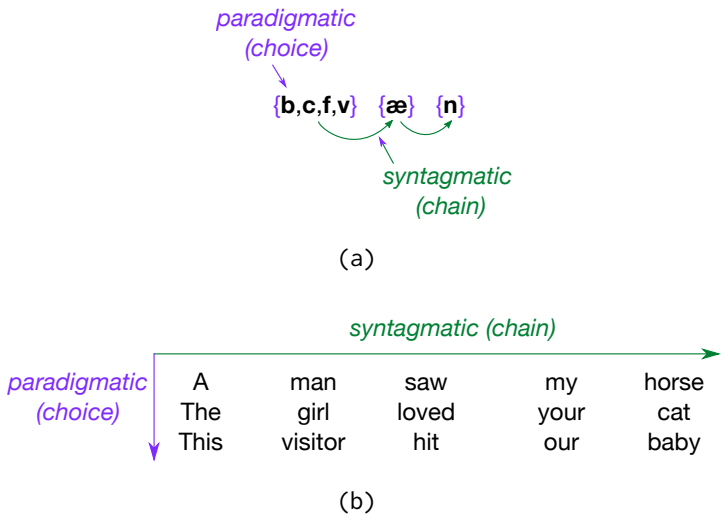


Figure 1.1: Syntagmatic and paradigmatic relationships on different levels of the abstract language system: (a) relationships on the sound system forming the words *ban, can, fan* or *van*; and (b) relationships on the textual/sentence level.

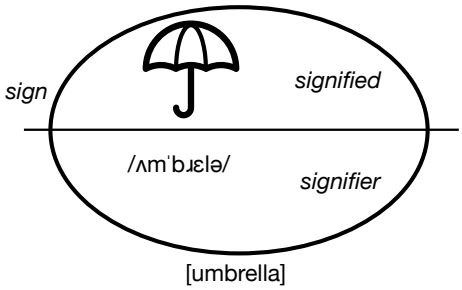


Figure 1.2: Illustration of de Saussures model of the linguistic sign, here on the example of the English word 'umbrella'.

the concatenation of all the sounds to form the final utterance, e.g., *b-a-n* (cf. Figure 1.1a)[3]. Figure 1.1b shows an example of syntagmatic and paradigmatic relation on the sentence level. Only the combination of *choice* and *chain* gives the completed system's specific construct its final incentive.

In de Saussure's model of the linguistic *sign*, i.e., his view of the nature of words, two constituents are inseparably connected: *a*) the *signifier* (signifiant), the form/sound, utterance or a sequence of sounds, and *b*) the *signified* (signifié), a concept/meaning, *function.* According to de Saussure, the signifier and the signified are interlinked and purely conventional within members of a language community, i.e., it is based on a common understanding and an inherent agreement (cf. Figure 1.2).

Generative Linguistics (Formalism) vs. Functional Linguistics (Functionalism) Another significant milestone was led by Chomsky (1957, 1965) and his influential idea of *generative grammar*. According to Chomsky, speakers of a language are able to build infinitely long sentences, of which many have not been ever produced before, so the "competence" of a speaker to form those sentences would require formal means to *generate* them, i.e., a *generative grammar*. Chomsky defines

3. Example adapted from (Kortmann, 2005).

the "competence of a speaker" as what a speaker knows about language and the world. A (generative) grammar is a formal system of rules and parameters that uses finite mechanisms to produce infinite structures. Because of his radical ideas and his influence in many scientific fields, i.e., linguistics, philosophy, mathematics, computer science, etc., Chomsky is considered one of the most influential individuals of the 20th century: "Chomsky is currently among the ten most-cited writers in all of the humanities (beating out Hegel and Cicero and trailing only Marx, Lenin, Shakespeare, the Bible, Aristotle, Plato, and Freud) and the only living member of the top ten." — (Pinker, 1994; p.23).

Chomsky had a very radical view on scientific methods, i.e., linguists should not rely on data to find regularities in language, but rather the goal should be to investigate and describe the "competence" of a speaker. This basically split the linguistic, scientific community into two camps:

1. *Rationalists* (symbolic): theories of language are built by applying logical thinking and human reasoning, and

2. *Empiricists* (stochastic): knowledge about a language comes from observations and stochastic generalizations.

Empiricists favored a model in which language and meaning can be described by means of statistical analysis following methods of *functionalism*. Functionalism is the linguistic research direction concerned with the two questions: *a*) Why is language what it is, and *b*) Why do speakers of a language prefer the use of one word over semantically equivalent other words? Functionalists, such as Bloomfield, thus acknowledge that language is highly situational and context-dependent and that language changes according to convention. In general, language, as a system, has thus to be investigated by its primary function it serves, namely communication, and this has to be done within its current setting (situation) and by its present purpose (e.g., informational, social). Bloomfield (1933) has thus proposed the "*discovery procedures*", which Chomsky argued so much against. One of those procedures is the *distributional analysis*, which gave rise to *immediate constituent analysis* through *constituency relations*, i.e., certain elements and their constituency relation counterparts follow a particular distribution. This initiated the field of *contextual theory of language*, where the *Distributional Hypothesis* (DH) was introduced by (Harris, 1951, 1954), and which was described best by Firth (1957; p.179) with his famous quote: "You shall know a word by the company it keeps.". With the rise of computational methods, the distributional hypothesis gained more and more traction and is nowadays the theoretical basis of modern *language understanding* systems.

Perspective

De Saussure's paradigmatic relations directly relate to Harris' distributional hypothesis, which we use extensively, and syntagmatic relations can be seen as the factor for the contextual disambiguation in the DH. We use grammar as a means to resolve meaning from long-range dependencies within longer sentences and use the DH to estimate meaning. The DH is also one of the major themes of this work; it is the basic theoretical foundation for the sense induction techniques introduced in Chapter 3, and it is implicitly used in contextual embeddings used in Chapter 4 and Chapter 5.

1.3 Semantics in the Linguistic Field

Semantics, within the linguistic field, is defined as the study of words, phrases, or longer textual units and their relationships with each other, which eventually form their individual meanings (Ogden and Richards, 1923). Semantics occurs on multiple levels of the language understanding system and can be roughly categorized into formal, lexical, and conceptual semantics.

1. *Formal semantics*, which uses mathematical tools (arithmetics) and logical constructs to analyze and reason about the information of the real world.

2. *Lexical semantics* is concerned with the relationship between words and their meaning within their naturally occurring context.

3. *Conceptual semantics* deals with the meaning of words and their possible interpretations before adding context, i.e., conceptual semantics studies the prototypical meaning of a word.

Sense (Thought) and Reference Gottlob Frege (1848–1925) questioned the notion of objects and concepts, their relationship within the language system, and their logical notation in his seminal work: *"Über Sinn und Bedeutung"* (Frege, 1892). The main question Frege considers is: "What is the identity between two objects/signs" (Frege, 1892). This is also commonly known as *Frege's puzzle to the law of identity*. For example, consider the phrases 'Venus is the morning star' and 'Venus is the evening star'. Both sentences claim that the term 'Venus' is identical to the terms 'morning star' and 'evening star'. It follows that the 'evening star' and the 'morning star' should be identical too, but according to Frege, they are not: one can be seen in the morning, and the other can be seen in the evening. Frege thus distinguishes between the *sense* of an object and its *reference*, i.e., in the example, 'Venus' can be considered as the thing both expressions 'morning star' and 'evening star' refer to, i.e., the sense is the same, but the reference is different and cannot be used interchangeably. Frege thus derived that the sense of a sentence, i.e., the thought, is its knowledge value, whereas the referent of a sentence is its truth value.

Ogden and Richards (1923) proposed in their work "The Meaning of Meaning: A Study of the Influence of Thought and of the Science of Symbolism" the *semiotic triangle*, also known as the *triangle of meaning*, or *triangle of reference*

(cf. Figure 1.3), which illustrates the usage of a word with respect to its intended meaning (thought) and the explicit object (referent). It can be roughly compared to de Saussure's model of the linguistic sign, where the *sign* is a composition of its utterance in the language system and the *thought* it is supposed to invoke. The semiotic triangle describes the relationship between a *symbol* or *word* that is used to describe a *thought* or *reference* that, in turn, refers to a specific *object* or *referent*. Consider as an example the referent being a concrete table the speaker or hearer is currently looking at, and which is invoking a thought ('that table') or a reference ('table number two'), which is symbolized by a word or any kind of symbol in the language system. Note that the symbol symbolizes the subjective and individual interpretation, which does not necessarily overlap with the correct referent. The triangle (in Figure 1.3) shows there's no direct relationship from a symbol to its referent it is supposed to stand for, without invoking the loose thought. Thus the intended meaning of the symbol might not be the perceived one, and the perceived meaning does not necessarily stem from the intended one, and thus one has to consider the possible ambiguity of an utterance. This happens particularly in situations where the context is not generally acknowledged by all participants of the communication act.

The triangle can represent the *composition* of a message by an author or the *interpretation* of a message by the recipient. Searle (1975) illustrated this setting of a *speech act* by introducing the notion of *direction-of-fit*, which is more generally used in philosophy than in speech act theory, which we cover here. The term describes the fitting of a name to an item and the fitting of an item to a name. Consider as an example the illustration in Figure 1.4 where the square represents two combined semiotic triangles, one for the recipient of a message (the 'reader') and one for the author of a message (the 'writer'). The two cases of interest are:

word-to-world fit: a WRITER retrieves a WORD for the WORLD, and

world-to-word fit: a READER retrieves the WORLD for a WORD.

For completeness, Searle (1975) also identified two more notable cases, the double fit, which is a declarative act and includes the definition of a word to the world, i.e., it is always a **world-to-word-to-world** fit, and the **empty-fit**, for expressions that don't contain any sentential fit.

Production and Comprehension That being said, it is important to ask the question *"How does linguistic meaning come into being?"*. According to Miller et al. (1960), a speech act is an *action*, i.e., it is goal-oriented, and it presumes that the speaker and the listener share a specific task that has to be solved by acts of communication. Three essential phases can be described by the intentions and properties of speech comprehension and production (Griffin and Ferreira, 2006):

1. Planning: forming a thought, the task is specified, its conditions in the instantaneous situation are studied, and a plan of action for solving the task is designed and gradually elaborated;

2. Execution: communicating, the plan is executed, and the transition is made from a purely intellectual activity to external activity;

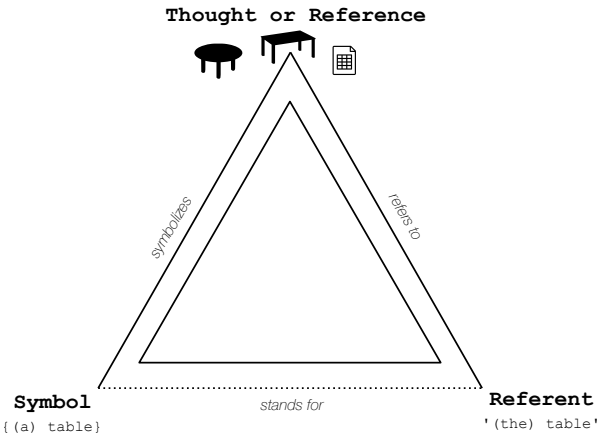


Figure 1.3: Excerpt of the the semiotic triangle as defined by Ogden and Richards (1923). `Referent` is the explicit object/thing we're talking about, `Thought or Reference` is the implied concept, and `Symbol` is the sign/word used to evoke the thought.

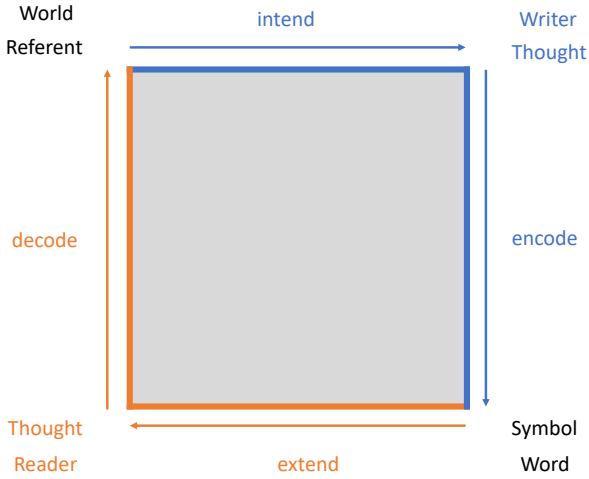


Figure 1.4: Illustration of the *direction-of-fit* with regard to a speech act, which involves the word-to-world and world-to-word fit. It contains two semiotic triangles, one for the author or writer of a message and one for the reader or recipient.

3. Control: receiving feedback, the result achieved by the action is compared with the goal set, and if the goal was achieved, i.e., the message and its information have been received, and a common understanding has been acknowledged, the action is considered completed; otherwise the subject needs to return to the planning phase and re-elaborate or refine the plan of action.

Meaning as Use Ludwig Wittgenstein (1889-1951) defined that a word's meaning and hence the message of a sentence depends on its use and can only be correctly resolved in a common (social) context (Wittgenstein, 1953). Speakers form an agreement on the situational context such that the meaning of a word or an expression can be resolved by means of a common reference. Wittgenstein introduces the notion of the so-called language game, which he uses as a metaphor for the communication act being a tool for solving a particular task, not the

communication act itself. It follows that communicative acts may be similar, but they are still of different use. Wittgenstein claimed, 'In most cases, the meaning of a word is its use', i.e., many arguments in a conversation can already be resolved by asking, 'Are we even talking about the same thing?'.

Perspective

We ground our work of computational unsupervised sense induction in Chapter 3 on De Saussure's notion of the linguistic sign, Frege's notion of sense and reference, and the notion of the semiotic triangle by Ogden and Richards, which cover the philosophical distinction of words and their activation in the human mind, i.e., what we think words mean. In Chapter 3, 4 and 5, we explore contextualized embeddings, which can be interpreted as an instantiation of Wittgenstein's notion of meaning as use. All of them define the notion of senses, which we cover as means for extrinsic evaluation in Chapter 2 and Chapter 3 for unsupervised sense induction using static and contextualized embedding approaches.

1.4 Modern Linguistics

Language Modeling and Information Theory Within the stochastic camp and also influenced by other scientific fields, such as electrical engineering and physics, another direction of computationally-oriented linguistics emerged. Here, the rise of *probabilistic language models* led to advancements in machine translation, speech recognition, and many more.

To measure the performance and success of a language model (LM), concepts from the field of *information theory* have been borrowed, for example, entropy, perplexity or information gain, which themselves have been borrowed from the field of thermodynamics (Shannon, 1948). Entropy is originally defined as a measure of chaos, whereas here, it is used implicitly for measuring the information content of a language – respectively a language model – and was first introduced to information theory and linguistics by Shannon (1948), who was also the first to measure the entropy of English.

With more computational power and easier access to data, *neural language models*, i.e., neural networks trained with language modeling objectives, such as *next word prediction*, lifted the performance of Natural Language Processing tasks to an industry-ready level. Here, neural language models are pre-trained and further employed as so-called *feature generators*, where the model either provides the learned static weights of the training vocabulary (static word embeddings, SWEs) or the model is applied to a sentence or sequence of strings, in a *prediction* or *fine-tuning* setup and the updated dynamic weights of the internal neural layers are then used as features (contextualized word embeddings, CWEs) for solving virtually any downstream task (see for example Devlin et al., 2019, sqq.).

Perspective

All our work is based on progress in modern linguistics. It provides us with the necessary tools to analyze large data collections, find new, hidden artifacts within a language, and help us, humanity, search and find information in vast collections of data such as the World Wide Web (WWW). Its presence is ubiquitous in everyday life; we, humanity, use those tools actively and sometimes even unknowingly. In Chapter 2, we employ statistical language models and perplexity for measuring domain affinity. In Chapter 3, we retrofit static word embeddings, created as a by-product from a neural language model, to static sense embeddings, which we use for word similarity tasks. We continue this line of research and further analyze more recent contextualized word embeddings, which come from training very large neural language models, and test their ability to implicitly encode senses (Chapter 3), their suitability to be used for information retrieval (Chapter 4), and their ability to be used for unsupervised modeling of semantic relations (Chapter 5).

1.5 Motivational Aspects

However good existing pre-trained models are nowadays, due to the ambiguous nature of language and because it is ever-changing and evolving, NLP systems continue to make errors, however small they might seem. This stems mainly from the complexity of ambiguities, which humans often can resolve easily because of our ability to reason within the situational context. In this dissertation, we highlight the importance of context and investigate methods to supply context on different levels. The work in this dissertation addresses two subject matters:

1. the enhancement of domain-dependent corpora to supply context on a vocabulary level, and

2. the analysis of recent contextualized word embeddings.

We emphasize that context matters; *a*) on the level of domain-dependent data, as well as *b*) more fine-grained on the representation level.

1.5.1 Domain-Dependent Corpus Enhancement

Context can be supplied on various levels, and the biggest issue is the technical feasibility and the algorithms to reason beyond a single source of data. Here, we address the issue of domain-dependent natural language processing. Nowadays, large collections of text corpora exist, which can be exploited and used to create NLP models for various downstream tasks. However, models trained on generic data often fail to produce precise results for domain-dependent tasks. To counter this issue, the models are often trained on data of a particular language domain, e.g., the biomedical domain. By this, models are presented with a different or limited vocabulary, with different sentence structures, and so on. This already defines context on the level of input data. However, data is usually limited for special domains. The research question we address in Chapter 2 can be formulated as:

How to enhance a given small corpus of documents by crawling only relevant parts of the world-wide-web, and what is the benefit of using domain data over generic data?

1.5.2 Semantic Structuring

Embeddings – static vectors, generated with matrix factorization or neural networks (NNs); or contextualized embeddings (CWEs), which are currently exclusively generated with NNs – are essential tools in natural language processing (NLP) for computationally representing elements of language such as words, sentences, phrases, documents, etc. Standard *vector space models* (VSMs) of words already provide inherent semantic properties since the features of a word depend on the context of its occurrence, and thus, they implement the *Distributional Hypothesis* (DH; Harris, 1954). The research questions we address in Chapters 3, 4 and 5 investigate the advantage of using contextualized word embeddings for downstream tasks compared to static word embeddings:

How can pre-trained language models be used to induce senses, and what is the benefit of contextualized embeddings over static embeddings for information retrieval and semantic structuring tasks?

1.6 Vector Representations & Embeddings

Since we use the term *embeddings* extensively in this dissertation, we introduce the most important concepts in this section.

Vectors are the computational representation of choice for lexical units (LUs), such as words, phrases, sentences, or entire texts, for the use in machine learning (ML) algorithms. The dimensions in these so-called feature vectors represent the features of a lexical unit. Each feature activates a non-zero entry in its respective dimension, and many options exist for what a good feature can be. In the distributional space, cf. the distributional hypothesis (DH; Harris, 1954), a feature is the context in which a word occurs. In one of the most straightforward cases, the context is defined as the frequency value of the co-occurrence of a word v with another word w when they appear in the same sentence, i.e., the *bag-of-words* (BoW) approach. The final model, when collecting all co-occurrences of all words in a corpus, is then also called a term-term matrix by sentence co-occurence, and it is potentially very large ($|V| \times |V|$ where V is the vocabulary) and very sparse, since word occurrences, as well as word-to-word co-occurrences, follow a power-law distribution, i.e., a few words often co-occur with many other words,

but many other words only co-occur with a few words (see for example Biemann et al., 2022). This 'standard' *vector space model* (VSM) of words already provides intrinsic semantic properties, just because the occurrence of a word is represented as a feature (context) of another word; thus the DH is applied. VSMs span many representations, *feature engineering* was once an art, where scientists were creative about finding the best features for representing lexical units, e.g., TF-IDF (term-frequency inverse document frequency) is another popular representation of documents or spans of text often used for information retrieval (IR). However, those huge but sparse vectors in the explicit space are more complex to work with computationally than lower-dimensional but dense vectors.

Embeddings (in NLP) are representations of the explicit huge vectors in a different space, typically a lower-dimensional dense space, where semantic properties and mathematical operations of the original vector space, such as similarities with respect to symmetry, reflexivity and transitivity are *a*) preserved, or *b*) even facilitated certain phenomena such as non-overlap in certain dimensions due to sparsity issues. Mathematically more formally: one space X is embedded in another space Y when the properties of X are the same as the properties of Y, e.g., natural numbers $\mathbb{N}$ are embedded in integers $\mathbb{Z}$, which are embedded in rationals $\mathbb{Q}$, which are embedded in reals $\mathbb{R}$ and so on. In our BoW example, vectors can be modeled using natural numbers, e.g., $\vec{v} \in \mathbb{N}^{|V|}$, but vectors in the embedded space are modeled using real numbers, i.e., $\vec{v} \in \mathbb{R}^d$, where the dimension d is much smaller than the dimension in the BoW space $d \ll |V|$. Estimating such representations has long been an active research area, and it continues to be, although aspects have changed, i.e., nowadays, embeddings are mainly estimated using deep neural networks (DNNs), and the art lies in engineering the network architecture of the respective layers as well as in their analysis.

Static word embeddings (SWEs): The history of word embeddings is vast, ranging from regression models such as WordSpace (Schütze, 1992b), and geometrical matrix factorization methods like principal component analysis (PCA; Schütze, 1992a) or latent semantic analysis (LSA; Landauer and Dumais, 1997), over to probabilistic topic modeling such as probabilistic latent semantic analysis (PLSA; Hofmann, 1999), or latent dirichlet allocation (LDA; Blei et al., 2003) to neural approaches such as skip-gram negative sampling (SGNS), continuous bag-of-words (cBOW), which are both available in the Word2Vec toolkit (Mikolov, Chen, et al., 2013), or global vectors (GloVe; Pennington et al., 2014), all of which build upon neural network (NN) architectures and which were a significant milestone in NLP progress. Since we make extensive use of SWEs from the Word2Vec[4] toolkit, we want to introduce it here schematically.

Word2Vec (Mikolov, Chen, et al., 2013) applies a neural language modeling approach, where the goal is to predict a word w_i at position i given its context c_i, in other words, the goal is to optimize the probability $p(w_i|c_i)$. In neural approaches to language modeling, weight matrices are learned as a by-product, representing the activations of the hidden units of the network. Those weight matrices are the

4. The trained embeddings for various corpora and the source code are available at https://code.google.com/p/word2vec/

lower-dimensional embedded representations of the vocabulary and are later used as so-called static word embeddings (SWEs). In traditional language modeling schemes, c_i is defined to be the sequence of $n-1$ words occurring directly in front of w_i, i.e., w_{i-n+1}^{i-1}, where w_k^l refers to the sequence of words from position k to position l. However, the definition of the context c varies for practical reasons when focusing on semantic word representations. See, for instance, (Riedl, 2016) for an overview of different context types. Mikolov, Chen, et al. (2013) introduced the SGNS (skip-gram-negative-sampling) architecture, where w_i is used as input and the surrounding words with window size m are used as optimization target, i.e., $\{i \neq j | w_{i-m}^{i+m}\}$. Accordingly, in the continuous bag-of-words (cBOW) model, a joint representation of context words is used as input to optimize the prediction probability of the current word w_i as the target. A projection matrix is learned during this process, which exhibits syntactic and semantic properties and has been shown to be beneficial in various NLP tasks. E.g., Mikolov, Yih, et al. (2013) showed that simple vector arithmetics, i.e., addition and subtraction, can be used to address analogies, i.e., linguistic regularities like in the famous example $\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} \simeq \overrightarrow{queen}$ are implicitly modeled. However, static word embeddings are estimated for the vocabulary, i.e., a word w is always represented by the same vector, regardless of its occurrence in a sentence; they are thus context-unaware.

Contextualized word embeddings (CWEs): The next major milestone in NLP progress, which provided even further improvements for NLP systems, even beating human performance for some tasks, came with the introduction of *pre-trained language models* (PLMs) for *transfer learning* (TL; Ruder, 2019; Ruder et al., 2019), i.e., *deep neural networks* (DNNs; Goodfellow et al., 2016) are first trained with a certain objective to solve a task A, and then task-dependent fine-tuned to solve other tasks different from A. Most of these neural language models are trained in a self-supervision scheme to jointly predict *a*) masked words within the sequence, and *b*) the next sentence. Such language models are called *masked language models* (MLMs) because their objective is different from traditional language modeling, which is estimated using an autoregressive objective. In masked language modeling, the task is to predict words in a cloze-like fashion: $p(w_i | w_1, \dots, w_{i-1}, \texttt{[MASK]}, w_{i+1}, \dots, w_n)$, i.e., given a sequence of words $w_1, \dots, w_n$, the task is to predict the word at position i which is masked with the special token `[MASK]` in the sequence.

One implementation of a masked language model, which we want to highlight here because we use it in multiple chapters in this work, is *BERT* (Bidirectional Encoder Representations from Transformers; Devlin et al., 2019), which implements the encoder stack of the *Transformer* architecture (Vaswani et al., 2017). The Transformer is based on multiple *self-attention* layers, which allow a token to access all other tokens in the sequence. Self-attention and thus the non-directionality of the language modeling objective results in word embeddings that are context-aware, i.e., so-called *contextualized embeddings* (CWEs). The core idea of contextualized embeddings is *compositionality*, i.e., to compose static word embeddings so that the outcome represents the meaning of a word in a context-dependent fashion. Other models exist that provide contextualization by employing variations of *recurrent neural network*s RNNs, such as *Flair* (Akbik

et al., 2018) or *ELMo* (Embeddings from Language Models; Peters et al., 2018).

Further performance gains can be achieved by increasing the number of parameters of a model, i.e., the trainable weights of a neural network and the amount of training data. Zhao et al. (2023) present a summary of so-called *large language models*, which are, metaphorically spoken, PLMs on steroids, i.e., PLMs of a considerable size, consisting of hundreds of billions of parameters. However, it is often economically not possible to efficiently work with LLMs because they are *a*) often not available due to proprietary access, and if they are available, they are *b*) too large to handle for most institutions since training and prediction requires multiple GPUs or even a computer cluster with multiple GPUs per cluster node.

2 Crawling / Corpus Expansion

2.1 Overview

It is commonly known that the quality of computational models increases when exploiting more data. Halevy et al. (2009) explained the effect of simple methods superseding more advanced and complex methods when using larger amounts of input data, which is also supported by Brants and Franz (2006)'s or Banko and Brill (2001)'s findings. With the increasing power of computing resources and algorithms to process more and more data in less time and more efficiently, the demand for large text collections grows. The web, as a vast and dynamic resource, is nowadays the main starting point for knowledge induction systems like *NELL*[1] (Carlson et al., 2010) or *machine reading*[2] (Etzioni et al., 2008), although its size is more or less constant for the past two years[3] its content is in a steady change.

Thus, a variety of generic linguistic corpora were compiled using the web as a resource. E.g., large pre-trained language models such as BERT (Devlin et al., 2019), T5 (Raffel et al., 2020), GPT-{1,2,3,4} (Radford et al., 2018; Radford et al., 2019; Brown et al., 2020; OpenAI, 2023), ELMo (Peters et al., 2018), etc. are trained using web size corpora; they usually use the *Common Crawl* corpus[4] as one element in their collection of training corpora. The Common Crawl corpus is a collection of raw HTML web pages, metadata, and extracted plain texts crawled from the web and provided roughly every month as a snapshot of about (currently) 60 TB of compressed data. The *WaCKy*[5] corpora (Baroni et al., 2009), which were created from web documents of a particular *top-level domain* (tld) and by filtering for a particular language of interest, for example, *ukWaC* was created by limiting a web crawl to the `.uk` top-level domain selecting only English documents are available for a variety of languages. Another research initiative that produces such

1. Never Ending Language Learner: http://rtw.ml.cmu.edu/
2. http://ai.cs.washington.edu/projects/open-information-extraction or http://openie.cs.washington.edu/
3. https://www.worldwidewebsize.com/: last accessed on March 2022
4. Common Crawl Project: https://commoncrawl.org
5. Web-as-Corpus Kool Yinitiative http://wacky.sslmit.unibo.it/

corpora is the *COW*[6] (corpora from the web) project (Schäfer and Bildhauer, 2012), which also provides compiled web corpora for different languages from particular top-level domains. In (Goldhahn et al., 2014), we crawl top-level domains for under-resourced languages and use the *WebCorpus*[7] toolkit (Biemann et al., 2013) for processing the collected web data in a distributed manner. *ClueWeb* (Callan et al., 2009), yet another corpus from the web, and the *Common Crawl* Project provide mainly unprocessed HTML data. Those are further refined in various efforts, e.g., in (Pomikálek et al., 2012; Buck et al., 2014; Panchenko et al., 2018).

However, the data is largely collected without a notion of topical interest, which makes sense when the focus is on properties of general language usage and not on particular domains or genres. If an interest in a particular topic exists, corpora have to undergo extensive document filtering with simple and/or complex text classification methods. This leads to a lot of downloaded data being discarded and lots of computational resources being unnecessarily wasted for the crawling process.

One approach to work around these issues is to use the *BootCat* method (Baroni and Bernardini, 2004), which collects, based on keyword lists, web documents by sending combinations of keywords to a search engine provider. The returned URLs (Uniform Resource Locators) are then filtered by some heuristics like blacklisting or whitelisting and then downloaded and processed such that HTML and boilerplate content is removed. The resulting corpus is domain-dependent but usually of limited size and the process requires the use of a search engine provider as a black box, which makes it dependent on *a*) the general availability of the service, and *b*) ranking of the results based on the provider's (possibly subjective) choice (Kilgarriff, 2007).

In (**Remus**, 2014) and (**Remus** and Biemann, 2016), we propose a tool for *focused web-crawling*, which makes efficient use of computational resources, since its purpose is to download mainly websites of interest, i.e., those that cover a certain topic of interest. The topic or domain of interest is defined by a statistical N-gram language model or sample texts that can be used to create such a language model. Search engine providers are not a required component of the system; however, in practice, search engine providers can be used to create an initial starting seed of URLs. In (Panchenko et al., 2016), we use the focused web crawler to enhance given domain-dependent corpora to create unsupervised topical taxonomies. And in (**Remus** et al., 2016), we present a robust tokenizer and POS tagging approach for the diverse input that the web provides, i.e., the different genres such as news texts or socially interactive communication. We run several experiments to show the effectiveness of the tools.

2.2 Focused Web Crawling

Focused (and general) web crawling use cases include the development of web search engines (Chakrabarti et al., 1999; McCallum et al., 1999), compiling large linguistic corpora (Baroni et al., 2009; Medelyan et al., 2006; Schäfer et al., 2014), support for digital libraries (Qin et al., 2004; Pant et al., 2004), machine learning

6. http://corporafromtheweb.org/
7. http://sf.net/projects/webcorpus

(ML) corpora for large language models (e.g., Devlin et al., 2019) and many more. Building language-specific web corpora usually starts by limiting a web crawl to a certain *top-level domain* (Biemann et al., 2013; Baroni et al., 2009), which we refer to as *bounded crawling*, whereas for search engines, it is more important to cover the entire web, hence *unbounded crawling* strategies are typically used. Once a use case grows more specific, e.g., the empowerment of *domain-specific* search engines, corpora, or digital libraries, the demand for saving precious time- and computational resources becomes more and more important, and focused crawling techniques are sought. The term 'focused (web-) crawling' (Chakrabarti et al., 1999) also known as 'topical crawling' (Menczer et al., 2004) refers to the process of crawling the web in a guided way with a focus on a specific topical content. The main difference of standard *web crawling* and the main challenge for the focused crawler is to decide which link to follow, i.e., which link might lead to the desired topical content and which not. This can also be re-formulated as a priority-oriented task, i.e., in which order should the crawler follow links before actually downloading the content of their respective destination, all of which happens during the actual crawling time (Chakrabarti et al., 1999). This can either be seen as a classification task (McCallum et al., 1999; Chakrabarti et al., 1999; Medelyan et al., 2006) with binary decisions (follow or not follow) or as a ranking problem with a priority queue where URLs with a high chance to lead to relevant content are prioritized.

Obviously, the main issue is to estimate the relevance of a yet unknown document with only a little prior information. Our approach differs from existing approaches in that we employ a *language model* and *perplexity* as a measure of relevance for a particular web page, whereas other approaches use features such as individual components of the URL itself, e.g., server, path, query strings, etc., the surrounding context of the extracted hyperlink, the text of the parent webpage, include lexical resources like *WordNet* (Fellbaum, 1998), and many more (Safran et al., 2012). One major advantage of the proposed methodology is the deliberate omission of negative instances for modeling. This being said, the method does not need labeled data (Blum and Mitchell, 1998), neither needs the focus to be predefined as a certain category in a predefined taxonomy (Chakrabarti et al., 1999), and it also does not require manually constructed lexical resources for feature extraction (Safran et al., 2012), but only operates on an initially provided corpus of plain texts, which serves as the domain definition. E.g., in one of our experiments, we used only one Wikipedia article as the domain definition.

2.3 Methodology

Web pages of a certain genre or domain use a certain vocabulary (Biber, 1995), and these web pages, in turn, link to other web pages of the same genre or domain. This oversimplifying assumption is typically exploited in focused crawling (Chakrabarti et al., 1999; Menczer et al., 2004; Safran et al., 2012; Medelyan et al., 2006). Our approach follows previous work and relies heavily on this assumption by using *statistical language models* (cf. Section 2.3.1) combined with *perplexity* as a measure of relatedness (cf. Section 2.3.3). The priority, i.e., the relevance of an extracted web link pointing to a topically relevant web page, is determined by the perplexity

value of its parent document.

2.3.1 Modeling Domain Specificity

Statistical language models (LMs) are a well-understood device in natural language processing. The techniques are successfully employed in many applications and downstream tasks, such as speech recognition (Kuhn and de Mori, 1990) or machine translation (Koehn et al., 2007). *N-gram language models*, as used in this work, are statistical models over short sequences of consecutive tokens called N-grams. The probability of a sequence of m words, e.g., a sentence, is computed as:

$$p(w_1 \dots w_m) \approx \prod_{i=1}^{m} p(w_i | w_{i-N+1}^{i-1}) , \tag{2.1}$$

where N is the order of the language model and $p(w_i|w_{i-N+1}^{i-1})$ is the conditional probability of observing word w_i given its history of N-1 words. w_i refers to the word at position i and w_k^l refers to the subsequence of words from position k to position l. A vast amount of techniques can be used to compute the conditional probability of an N-gram; see, for example, (Chen and Goodman, 1998) or (Goodman, 2001) for an overview. Most of those methods are derived forms of the simple *maximum likelihood estimate* (MLE):

$$p(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-N+1}^{i})}{c(w_{i-N+1}^{i-1})} , \tag{2.2}$$

where $c(w_k^l)$ is the frequency of the subsequence w_k to w_l in a training corpus. Since this model does not handle unseen N-grams or words, a number of different techniques were proposed which support the conditional probabilities for unknown sequences and words. Kneser and Ney (1995) introduced one of the most successful language models, which, despite its age, still exhibits good performance, especially for small corpora. Kneser and Ney apply interpolation techniques based on the number of distinct words that follow an N-gram $N(w_k^l\bullet)$, the number of distinct words that precede an N-gram $N(\bullet w_k^l)$, and the number of distinct word combinations surrounding an N-gram $N(\bullet w_k^l\bullet)$. The probability of an is then computed as:

$$\begin{aligned} p(w_i|w_{i-N+1}^{i-1}) = & \frac{\max\{0, D - c(w_{i-N+1}^{i})\}}{c(w_{i-N+1}^{i-1})} \\ & + \frac{D \times N(w_{i-N+1}^{i-1}\bullet)}{c(w_{i-N+1}^{i-1})} \\ & \times p_{smooth}(w_i|w_{i-N+2}^{i-1}) \end{aligned} \tag{2.3}$$

with D being a discount factor estimated from a held-out test split, and

$$p_{smooth}(w_i|w_{i-N+2}^{i-1}) = \frac{N(\bullet w_{i-N+2}^{i})}{N(\bullet w_{i-N+2}^{i-1}\bullet)} \tag{2.4}$$

being defined as the back-off probability. More details regarding the Kneser-Ney model can be found in (Kneser and Ney, 1995) or (Chen and Goodman, 1998)

who, propose an improved version, the so-called *Modified-Kneser-Kney* method, which uses different discount values for the number of distinct words following, preceding or surrounding an N-gram. In this work, we follow the suggestions of Chen and Goodman (1998) and fix the discount factor D to $n_1/(n_1+2n_2)$, where n_i represents the number of unique N-grams with count i, and apply interpolation or smoothing to uni-gram probabilities. We also use a different discount factor for every N-gram cardinality. In most of our experiments, we used a 5-gram model[8]. Also, we filtered numbers and punctuation characters and considered only sequences with a minimum length of five words. For segmentation purposes, i.e., for splitting a character sequence into sentences and tokens, we used the LT-SEG segmenter that we introduced in (**Remus** et al., 2016).[9] We developed this fast and robust segmenter, which is based on processing a set of rules, with the goal to segment diverse, web-based documents, that might also come from short social media texts, like Twitter[10] or Reddit[11].

Nowadays, neural variants of language models, more specifically, so-called *neural language models*, e.g., as first proposed by Bengio et al. (2003), based on neural network architectures, or *mixture language models* (Tan et al., 2012) are more popular due to their pre-training objective and general applicability for virtually any downstream task (e.g., Devlin et al., 2019). We refrain from following this line of research in the focused web-crawling setup for reasons of computational and conceptual complexity. Further, those models usually require lots of data to achieve a decent performance, which is not in line with our corpus enhancement setup, where the initial domain-defining corpus is defined to be of a rather small nature.

2.3.2 From HTML to Text

Structural elements of a web page, such as headers, footers, menus, etc., are rather undesirable since they occur on many web pages and usually do not provide valuable information. One of the processing steps is extracting important text from the bulk of structuring HTML code, javascript, CSS, etc. of a web page. We use the *boilerpipe* toolkit[12] (Kohlschütter et al., 2010) for *HTML stripping* and *boilerplate removal*. This step removes all parts of a web page that redundantly occur on many web pages, like navigational elements, advertisements, etc. For textual pre-processing, we filtered numbers and punctuation characters and considered only sequences with a minimum length of five words.

2.3.3 Perplexity Measure

We propose *Perplexity* (*PP*) as the measure of compatibility of the language model to an arbitrary text extracted from a web document. Perplexity measures the *surprise* of an underlying probabilistic model that is being confronted with test

8. 5-grams were chosen because experience in preliminary experiments yielded better results than other N-gram models.

9. The segmenter is available as an open source project under a permissive license: https://github.com/tudarmstadt-lt/seg.

10. https://twitter.com/

11. https://reddit.com/

12. https://code.google.com/p/boilerpipe

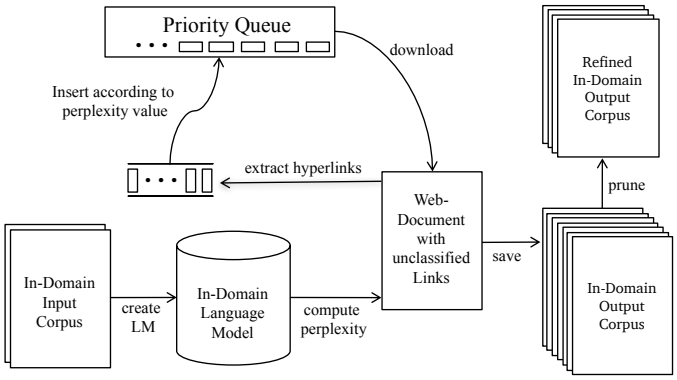


Figure 2.1: Schematic overview of our focused crawling process.

samples X, where we define X to be the set of N-grams extracted from the usable text of a web page. Perplexity, often used to evaluate language model performance, is defined as

$$PP(X) = 2^{H(X)} \text{ , where} \tag{2.5}$$

$$H(X) = -\frac{1}{|X|} \sum_{x \in X} \log_2 p(x) \, . \tag{2.6}$$

$H(X)$ is the cross entropy, and $p(x)$ refers to the probability of a particular N-gram in the set of test N-grams X.

URLs are prioritized based on the perplexity value of the parent document, i.e., the document it was extracted from. We maintain a priority-queue-like structure of URLs, which have been collected so far, and process this queue in increasing order, i.e., a lower perplexity score is preferable over a higher value. URLs extracted from documents that align better with the language model are thus considered for download before others. Figure 2.1 shows a schematic overview of this focused crawling procedure. Note that our proposed method does not rely on any access to knowledge resources like taxonomies or WordNet (Safran et al., 2012), nor does it rely on positively and negatively labeled HTML documents or web links (Blum and Mitchell, 1998; Chakrabarti et al., 1999; McCallum et al., 1999). The target domain is also not pre-defined and can deviate from ODP[13] categories (Chakrabarti et al., 1999).

2.3.4 System Architecture

The open source crawler software *Heritrix*[14] (Mohr et al., 2004) forms the basis of our focused crawler. Following Baroni et al. (2009), Schäfer and Bildhauer (2012), and Callan et al. (2009)[15], we use Heritrix (Mohr et al., 2004) as the base crawler, because it provides a well-established and sophisticated crawling framework and is extensible due to its modular design.

13. Open Directory Project, `http://dmoz.org`

14. http://crawler.archive.org developed and used by the *Internet Archive Project*: https://archive.org

15. For ClueWeb12 (http://www.lemurproject.org/clueweb12.php/).

Heritrix's architecture follows suggestions by Manning et al. (2008) and uses one queue per server; the set of all queues is called *frontier*. Our priority scheme includes URL and server queues such that the highest priority of a member URL determines the priority of a server queue. However, for more efficiency in a practical application, we introduce a bucketing strategy, which splits the perplexity range into three buckets called HIGH, MEDIUM, and NORMAL in Heritrix terminology. URLs in higher prioritized buckets will be downloaded before others, even if this delays the overall crawling process. The boundaries, i.e., the perplexity value ranges for the buckets, must be assigned by the user and can be determined by running short test crawls. Using this strategy, it is possible to dynamically change the behavior of the crawl during runtime by ignoring or reconsidering URLs from lower prioritized buckets. In our experiments, we did not change the strategy after a crawl started since this would lead to non-comparable results. URLs within the buckets are still ordered by priority, and for further efficiency reasons, we include so-called *budgeting costs*. Each per-server queue in the crawler's frontier has a certain budget of being processed by a worker, i.e., URLs being downloaded. The lowest budget a queue can have is 1, i.e., URLs will be processed with the lowest possible priority in the current bucket. Note that a possible option is to remove the server once its budget has been emptied, i.e., no further URLs from that server will be downloaded. We did not use this option, though. Lower priority, respectively, higher perplexity values will have higher costs for the per-server queue, which means that if a particular server provides many irrelevant documents, it will have a lower priority of being processed, although some individual documents might be relevant. For this, the perplexity value which was assigned to a URL is mapped to the interval $[0, 127] \in \mathbb{N}$ (a technical requirement for Heritrix) and used as a cost factor. After following a URL, the per-server queue *budget* is reduced by the assigned cost of the URL's underlying web document. Servers with many 'bad' URLs will exhaust their budget earlier than 'good' servers and will thus be considered less often, even if some URLs are still queued with a high priority.

Note further that the general crawler architecture is designed to process and download web pages in parallel. A URL's priority value is superseded by *politeness* settings regarding queued servers. *Politeness* is a self-imposed waiting time between consecutive requests to the same server. This is an essential setting since website administrators often block IP addresses of automated systems for security reasons. Imagine that a web server has to decide between automated crawling for a good cause and, e.g., a DDOS (Distributed Denial-of-service) attack, where random requests are made to force a server downtime. For a crawler to be considered a 'good' crawler, consecutive requests to the same server should be reduced to a minimum.

Parallelism and politeness, are advantageous and necessary design principles for modern web crawler systems (Manning et al., 2008; pp. 451-453), but unfortunately they interfere with the priority ordering in a focused crawling setting. Parallelism defines how many server queues are processed from the frontier, i.e., only entire servers are processed in parallel. As an example, consider three URLs; two URLs refer to one server and have a higher priority than the remaining URL, which refers to another server. Since by default, the crawler fully respects the

`robots.txt` specification[16] and is configured to be polite concerning the number of consecutive requests to the same server, the lesser prioritized URL can, and will, be considered before the second higher prioritized URL because the first request will impose a delay to the second request to the same server. In our evaluation, we show the advantage of our approach in a single-threaded and blocking environment, although this is not useful in practice. Preliminary experiments also show cleaner topic-dependent corpora when parallelism is deactivated for the price of longer waiting times to reach the same amount of data (if even possible) as opposed to a larger parallelism value.

2.4 Evaluation

Automatic techniques for measuring the relevance of webpages regarding some constraints have to be applied post-hoc because manual labeling of webpages is simply non-economical in terms of manpower. Even evaluating only a portion of the web is too much effort or does not mirror real-life conditions.

Chakrabarti et al. (1999) treat focused crawling as a binary classification task where URLs are classified as pointing to topically related webpages using several features of the source webpage. Features are, for example, the structured parts of a URL, such as domain, server, path, etc., the anchor text where the web link is embedded in the source document and its surrounding context. Gold labels for training the classifier are extracted by choosing a certain category from the Open Directory Service (ODP)[17]. In (Menczer et al., 2004), an evaluation framework is presented, which operates only on the set of ODP pages. Relevance is measured directly from the ODP hierarchy, such that recall and precision can be exactly calculated. Also, a lexical similarity score is computed based on the cosine similarity to previously extracted topic definitions for collected pages that are not in the ODP, e.g., links from ODP leaf pages. Safran et al. (2012) used the set of URLs retrieved from a search engine for evaluation. However, this strategy is non-deterministic because of the black-box behavior of web search engines. A different approach is presented by Meusel et al. (2014), who focus on structured data in the so-called *semantic web*. Evaluation is done by considering webpages that contain structured data as relevant. Schäfer et al. (2014) presented a focused crawling approach, focusing on linguistically clean data, i.e., webpages that can be used to construct web corpora. The very same function for cleaning a webpage is applied to assess relevance.

Note that we do not attempt to re-implement other focused crawling strategies because they rely on expensive supervision or are limited to domains defined by ODP categories. Here, the primary goal is to enhance a small given text corpus by crawling the web for more texts of the "same kind". We evaluate the system in intrinsic and extrinsic settings.

16. http://www.robotstxt.org
17. Open Directory Project: https://www.dmoz.org/

2.5 Intrinsic Experiments

In this section, we conduct two intrinsic experiments to compare our focused crawling strategy to the standard breadth-first web crawling strategy:

1. focused vs. non-focused small scale, single-threaded crawls, limited to seed websites, and
2. focused vs. non-focused large scale, parallelized crawl for the German educational domain.

The notion of 'non-focused' is the default breadth-first-search-like crawling strategy in Heritrix.

For the first experiment , we defined four seed URLs, coming from two different languages and topical domains, i.e., a pair of URLs is either the same language or the same topical domain, but not both. We chose the topics *cats* and *technology* in English and German. The web crawls are limited to the websites defined by the seed URLs, where the assumption is that those websites are comprised of more web pages from the respective topic. We abort the crawl after collecting 100 documents in a non-focused setting for reference purposes and in the following focused-crawling settings: We created two language models, one from the English Wikipedia article for *cat*[18] and one from the German Wikipedia article for *Hauskatze*[19] (eng. *domestic cat*) and initialized two focused crawls for each language model. We then count the number of documents that were downloaded from each of the website servers. We hypothesize that the focused crawls download more documents from the websites corresponding to their domain and language definition.

For the second experiment , we conduct a more extensive crawl and collect roughly 500GB of HTML data on the *German educational domain* in a non-focused and focused setting. The initial domain defining corpus is provided by Nam et al. (2014), and its size is around 800K unique sentences. We split the corpus into two equally-sized *training* and *test* sets, using the training set for initialization of the language model and the test set for testing the crawler's performance after the crawl has ended. The language of the original data is mainly German but contains small amounts of other languages. We do not explicitly apply language filtering since the number of languages other than German is negligible, and the small impurities also test the robustness of the proposed method. The distribution of languages in the corpus is listed in Table 2.1.

We then build different language models using the training set plus the cleaned plain text data from the crawls. This is done in intervals after collecting certain amounts of data during the crawl. At each interval, the resulting language models are evaluated using the test set. Further, we handled sentence de-duplication such that each sentence occurs only once for training the individual language models.

For evaluation, we calculate perplexity of the language model trained on the aggregated corpora on the test set. We hypothesize that a more focused crawl

18. http://en.wikipedia.org/w/index.php?title=Cat&oldid=651849595
19. http://de.wikipedia.org/w/index.php?title=Hauskatze&oldid=139331448

Table 2.1: Distribution of languages in % in the train and test set as well in the focused (f) and non-focused (nf) crawl.

Language	*train*	*test*	f	nf
de	96.81	96.83	92.31	15.51
fr	0.57	0.56	0.62	3.09
en	0.00	0.00	4.50	73.19
nl	0.02	0.02	0.22	0.55
es	0.00	0.00	0.26	1.58
it	0.01	0.01	0.32	1.16
other	2.59	2.58	1.78	4.91

Table 2.2: Number of downloaded web pages for a non-focused crawl and two focused crawls based on an English and German language model for the domain 'cats'. The crawls were bound to the given websites and limited to 100 documents in total.

LM	catchannel.com	techcrunch.com	meine-katze.de	heise.de
–	27	25	25	23
(en) Cat	93	2	3	2
(de) Hauskatze	1	1	97	1

lets the perplexity value decline faster. We manually selected about 20 seed URLs, which refer to web pages related to the German educational domain. The maximum perplexity score achievable for the language model on the training data is around 10^7, which happens only for documents with lots of unknown words. We discard links from documents with a perplexity larger than 10^5. The crawls are not bound by any other limitation than perplexity scores. Hence, webpages are collected from arbitrary top-level domains, another advantage over corpora created with top-level domain crawling (Goldhahn et al., 2014).

2.6 Results Intrinsic Evaluation

Results of the first experiment, shown in Table 2.2, indicate that the focused crawler is able to focus on the specified topic. As expected, the non-focused crawl collects documents from all four websites in equal quantities. On the other hand, the focused crawls mainly collect documents from servers that correspond to their topic definition. E.g., the focused crawl based on the English Wikipedia article for *cat* downloads most documents from `catchannel.com`, a website containing cat-related content in English, and the focused crawl based on the German Wikipedia article for *Hauskatze* downloads most documents from `meine-katze.de`, a website containing cat related content in German. The non-focused crawl, however, downloads equally many documents from all seed URLs

(catchannel.com, techcrunch.com, meine-katze.de and heise.de).

The second experiment on the large-scale German educational crawl provides more insights into the behavior of parallelized focused web crawls. The crawler was started with 16GB of memory and 25 threads, which download and process URLs in parallel. Politeness settings were applied, and the robots.txt was respected, possibly leading to incompatible results. We experienced an average download rate of 2 MB/s and a maximum speed of 8 MB/s. After the first 24 hours, the crawler collected roughly 200GB of raw HTML data.

2.6.1 Adapting to Language

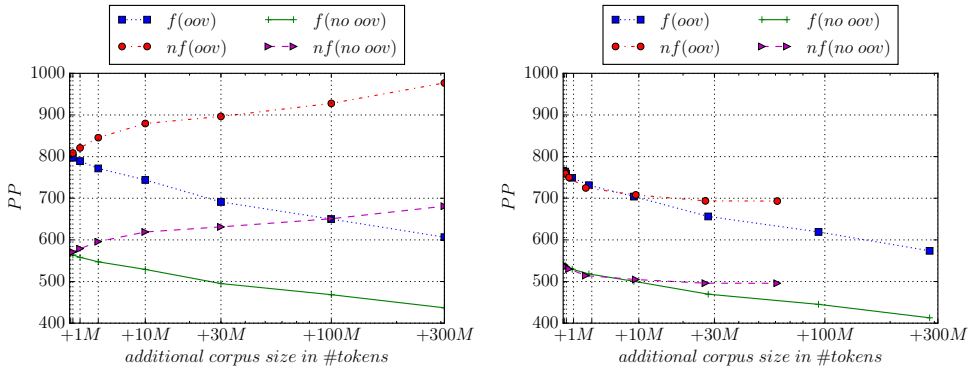
Because the crawler is generally un-bound, we collect URLs from a variety of top-level domains and also documents containing texts from different languages, something which obviously happens to a much larger extent in the non-focused setting (cf. Table 2.1). English is the primary interfering language since it is also the prevalent language used in the web. Although all seed URLs refer to domain-related German web pages, the non-focused crawl mainly collects web pages in English. The focused crawl still collects roughly equal proportions of languages as in the training corpus. This fact strengthens the point that an N-gram language models and perplexity is also a useful method for putting the focus just on a particular language. However, below, we show that the proposed approach is even able to capture domain aspects.

2.6.2 Adapting to Domain

As described in Section 2.5, we built separate language models at intervals, i.e., after certain amounts of data have been collected. The perplexity value on the held-out test set decreases with increasing corpus size in the focused crawl and increases with increasing corpus size in the non-focused crawl (cf. Figure 2.2a). While the former is expected, the latter seems surprising since the assumption might be that perplexity decreases with more data but at a much slower rate. However, due to the fact of incorporating a significant amount of non-target language data in the non-focused crawl, an overwhelming amount of out-of-vocabulary words are introduced, which eventually increases the perplexity of the entire set.

Since German is the prevalent language in our test corpus, with roughly 96%, we re-evaluated the test set perplexity for the same chunks of data by selecting only German documents.[20] Figure 2.2b shows that the focused crawler can harvest language and domain-relevant documents throughout the crawl; only a tiny fraction was discarded due to non-target language data, whereas almost half of the downloaded data was discarded for the non-focused crawl. When only considering German documents from the crawled data, the focused crawl yields consistently lower perplexity values; this difference increases as the crawl progresses. However, while more data is collected, the fractional amount of relevant / German vs. irrelevant / non-German data becomes even more prominent. I.e., after downloading 300M tokens, the unfocused crawl's usable German data

20. We use JLani from the ASV-toolbox (Biemann et al., 2008) for automatic language identification.



(a) (b)

size downloaded		100K	300K	1M	3M	10M	30M	100M	300M
size (*de*)	f	87K	267K	868K	2.66M	9.12M	27.78M	93.8M	277.30M
size (*de*)	nf	34K	95K	216K	558K	2.31M	9.45M	26.82M	61.12M

(c)

Figure 2.2: Perplexity on the test set by crawl size for German educational data (a) and crawls filtered for German (b) comparing focused(f) and non-focused(nf) crawling. Perplexity is measured on the test set, where *out-of-vocabulary* (OOV) words based on the train set are considered (*oov*) or removed (*no oov*). The corpus size is given in terms of the number of additional tokens for the training set. Table (c) shows the absolute number of tokens of German data vs. downloaded data for both crawls.

amounts to 61M tokens, and the focused crawl's usable data yields 277M tokens, which increases the usable harvest quantity by a factor of over 4.5.

2.7 Extrinsic Experiments

In this section, we evaluate the focused crawling application in extrinsic downstream task settings. We evaluate the crawler's performance on three topical domains and use the task of inducing in-domain taxonomies as an example application for the purpose of the extrinsic evaluation. Inducing *knowledge bases* from text is essential for knowledge management applications such as reasoning over facts and the new discovery of the like. In (Panchenko et al., 2016), we present TAXI, a semi-supervised system for taxonomy induction purely from texts, which makes use of additionally crawled data and is applied to in-domain data.

2.7.1 Taxonomy Induction

Algorithms, like the ones presented in TAXI, and applied for in-domain tasks, often still lack the desired input data size to create models that reasonably assess in-domain tasks.

Taxonomy induction is the task of inducing a taxonomy from text. *Taxonomies*, *ontologies*, *knowledge graphs* or, more general, *knowledge bases* are means for structuring information with application values in exploration or higher-level downstream tasks involving reasoning (Sowa, 2000), e.g., *question answering* or *entity linking*. In particular, a taxonomy can be seen as a hierarchical subset of

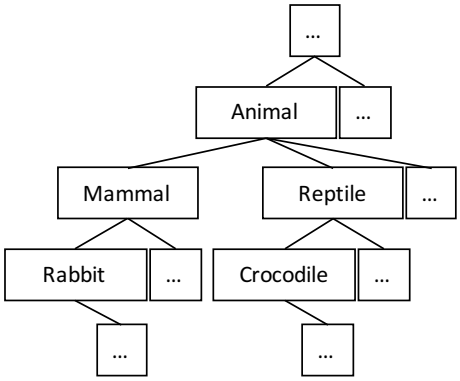


Figure 2.3: Example of a very simple taxonomy.

an ontology, containing only entities and relations that form tree-like structures between concepts with *is-a* resp. *subclass-of* relations. In linguistic terminology, a broader term is called a *hypernym* of a more specific *hyponym*; the relation is then *hypernymic*/*hyponymic*. Taxonomies are used in many fields, but the probably best-known example is the taxonomy used in biology to describe the biological ecosystem. Figure 2.3 depicts a simple taxonomy, in which *a rabbit* `is a` *mammal*, a *mammal* `is an` *animal*, a *crocodile* `is a` *reptile*, and a *reptile* `is an` *animal* too, among other instances and relations. Reasoning using the transitivity property then allows to infer that a *rabbit* is also an *animal*.

Usually, the information in such taxonomies is hand-crafted, thus expensive to create and maintain in terms of manpower. Hence, an active field of NLP research is concerned with automatic methodologies to build or extend taxonomies or knowledge bases from textual data without human assistance. Also, unsupervised or semi-supervised methods for taxonomy induction typically rely on a lot of text data. Needless to say, the textual domain dominates the topic of the concepts in the taxonomy. Many methods exist that use the web as a resource for such data simply because of the necessity of large amounts of data. Since the scope of the taxonomy depends on the scope of the used data and because taxonomies are often created using web data, most unsupervised taxonomies define a general domain. General domain corpora often miss important concepts if specific in-domain knowledge is required. Here, we use our *focused crawling* approach to collect in-domain data to induce *in-domain taxonomies*.

Inducing taxonomies by extracting taxonomic relationships from text is a well-known challenge; see, e.g., (Biemann, 2005) for a survey. Hypernymic relations can be extracted using various methodologies, e.g., by exploiting lexical patterns (Hearst, 1992; Oakes, 2005), to more complex statistical techniques (Agirre et al., 2000; Ritter et al., 2009) or methods based on neural methods. Snow et al. (2004) use a distant supervision approach where known hypernyms are selected, and parse trees are used as features to train a hypernym classifier, which is then used to extract new hypernyms. Also, the features can be interpreted as patterns, and highly activated features can be used as patterns for hyponym-hypernym discovery.

In (Yang and Callan, 2009), a semi-supervised taxonomy induction framework was presented, which integrates co-occurrence, grammatical dependencies, lexical-

syntactic patterns, and other features to learn an ontology metric, which is calculated in terms of the semantic distance for each pair of terms in a taxonomy. Terms are incrementally clustered on the basis of their ontology metric scores. In their experiments, Yang and Callan (2009) only use the word senses within a particular WordNet sub-hierarchy to ensure there is no ambiguity. Evaluation was done by replicating hypernyms of 12 *WordNet* sub-hierarchies, like 'building', 'milk', and 'meal'. Snow et al. (2006) perform incremental construction of taxonomies using a probabilistic model. They combine evidence from multiple supervised classifiers trained on large training datasets of hyponymy and co-hyponymy relations. The taxonomy learning task is defined as the problem of finding the taxonomy that maximizes the probability of individual relations extracted by the classifiers. However, rather than learning a new taxonomy from scratch, this approach aims at attaching new concepts under the appropriate nodes of an existing taxonomy, e.g., WordNet. A related, weakly-supervised approach aiming at categorizing named entities and attaching them to WordNet leaves was proposed by Pasca (2004). Other approaches mostly use formal concept analysis (Cimiano et al., 2005), probabilistic and information-theoretic measures to learn a taxonomy from a folksonomy (Tang et al., 2009), Markov logic networks, and syntactic parsing applied to domain text (Poon and Domingos, 2010).

Kozareva and Hovy (2010) start from a set of root terms and use Hearst-like lexico-syntactic patterns to harvest hypernyms from the Web. For example, running a search pattern like '* such as lion and cat' using a web search engine, 'feline', can be obtained as a new intermediate concept. The extracted hypernym relation graph is subsequently pruned. Bordea et al. (2015) presented the first shared task on Taxonomy Extraction Evaluation (TExEval) focusing on taxonomies in different domains. The challenge was continued in (Bordea et al., 2016) and the TAXI system, which uses our focused crawling approach, ranked first (Panchenko et al., 2016). Mainly because TAXI provides a relatively simple and open-source solution to in-domain taxonomy induction, and since the challenge's evaluation framework is readily available for applicability, it makes it a good candidate for the purpose of evaluating the quality and usability of focused crawling results. Since we use TAXI for extrinsic evaluation, we introduce the system in more detail in the next section.

2.7.2 The TAXonomy Induction system (TAXI)

The TAXonomy Induction system (TAXI) by Panchenko et al. (2016) is based on two principles: scalability and simplicity. The general idea of TAXI is to receive a set of domain-defining terms as input, processes text corpora, and output a taxonomy. It is based on the following four general steps:

1. Domain-specific corpora are collected and merged with general-purpose corpora (e.g., with texts from Wikipedia).

2. Candidate hypernyms are extracted based on substrings and lexico-syntactic patterns, such as defined by (Hearst, 1992)

3. Candidates are then pruned, such that each term has only a few salient hypernyms

4. Optimization of the overall taxonomy structure, such as removing cycles and inserting links for disconnected components, is the last step in this procedure.

In (Panchenko et al., 2016), we argue that it is more important to be able to process large input data rather than performing complex extraction inference on little data. The system takes a text corpus as input and outputs a topically relevant taxonomy defined by the corpus. In our case, we limit the set of corpora to the singleton set of each crawled corpus and one general corpus. Having the data fixed, TAXI induces a taxonomy in 4 steps:

1. candidate based substring matching for seed terms,
2. corpus-based pattern matching, for instance, generation,
3. filtering of noisy relations by means of statistical measures, and
4. final taxonomy construction by removing graph cycles, etc.

For substring matching, the following score is applied: Consider the (possibly multi-word) candidate expressions U and V, where $|V| >= |U|$, i.e., V is longer than U, the hypernymy score $\sigma(U, V)$ is then calculated as:

$$\sigma(U,V) = \begin{cases} \frac{|V|}{|U|} & \text{if } |U| \geq 3 \text{ and} \\ & \quad \{u_l \in U, v_k \in V \ : \ \exists \, i \mid u_{1:|U|} \sim v_{i:|V|}\} \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

which is intuitively described as: The score is larger than one iff the expression U is at least three characters long and it matches a subsequence at the end of term V. Consider as an example the hyponym/hypernym instance 'apple tree' `is a` 'tree'. In some cases, $\sigma(U, V)$ is changed such that U must match the beginning of V, e.g., for French expressions or English expressions containing a preposition, for example, in 'sandwich *with* ham'. This method typically provides high precision results, although false positive examples will naturally occur, for example, 'money laundry' `is a` 'laundry' would be a wrong prediction. However, those instances are much rarer and are accepted mistakes. Despite its high precision, or maybe even because of it, the substring method is not applied in our scenario because it is corpus independent and only considers seed terms. While it might improve final results for each tested corpus equally, we do not want to interfere with the independent evaluation of the downloaded corpora. We want to measure the importance of the focused corpora, hence we are ignoring the substring matching method.

In the corpus-based method, candidate hypernyms are extracted via *lexico-syntactic patterns* à la (Hearst, 1992), often also referred to as *Hearst-patterns*. We follow TAXI and rely on the PatternSim (Panchenko et al., 2012) implementation.[21] To reduce the number of noisy relations, e.g., wrongly inferred hypernym relations in both directions, an asymmetric pattern-based hypernymy score between candidate terms is applied. This score is also neglected in this work since we are evaluating only one corpus collection at a time. Further pruning strategies on

21. https://github.com/cental/PatternSim

loose relation candidates are applied as detailed in (Panchenko et al., 2016). Finally, the full taxonomy graph is built by removing cycles and connecting disconnected components to the root node (Faralli et al., 2015).

2.7.3 Evaluation Procedure

Although we extrinsically evaluate the crawler's performance here, we actually test the crawler's performance in a two-fold setup, intrinsically by data analysis and extrinsically by measuring the performance of the tasks using the data. First, we present the perplexity values during runtime and show that the crawled corpora indeed improve over time. We do this again by creating different language models at different intervals and measuring perplexity regarding the test data. We then test the crawler's performance extrinsically using the task of in-domain taxonomy induction. We use three different topics from the trial dataset provided by the SemEval challenge on taxonomy induction (TExEval; Bordea et al., 2015). The challenge provides the input data and the evaluation framework for the automatically generated in-domain taxonomies:

1. artificial intelligence (AI),
2. plants,
3. vehicles.

For each domain (topic), a list of terms is provided that defines the domain of interest. The lists contain single word and multi word expressions (MWEs), e.g., 'neural networks', 'apple tree', etc. Our focused crawler, which we named TopiCrawler, takes as input:

1. an initial language model or a text corpus that can be used to train a language model, and
2. seed URLs as the crawler's starting point.

Similar to Panchenko et al. (2016), we use articles from the topic's respective Wikipedia category for initialization, i.e., we use them for training, and a random subset of articles from each subcategory of a particular category is retrieved and used for testing. In order to generate the seed URLs, we use web queries as used by the *BootCat method* (Baroni and Bernardini, 2004). BootCat takes as input a list of terms, creates tuples, triples, quadruples, or more, with random permutations of the terms, sends the tuples/triples/quadruples/... as a query to a web search engine, and returns a list of URLs. We generated $1,000$ random query triples per domain and limited the search result to the top 10 hits. An overview of the initial data is listed in Table 2.3.

2.7.4 Intrinsic Evaluation: Crawled Data

In order to test the language model capabilities, we chose to evaluate different values of n, i.e., different sizes of language models. We tested $n = 1$, which represents a simple bag-of-words (bow) approach vs. $n = \{2, 3, 5\}$, which take the

Table 2.3: Data as provided by the TExEval organizers and collected by BootCat.

Domain	*ai*	*plants*	*vehicles*
Terms	2,218	513	93
Queries	1,000	1,000	1,000
URLs	1,605	6,620	7,414
#Articles (train)	263	85	49
#Sentences (train)	11,371	4,915	1,696
#Tokens (train)	212K	82K	30K
#Articles (test)	250	250	250
#Sentences (test)	12,901	5,753	14,473
#Tokens (test)	212K	96K	294K

Table 2.4: Collected data using different crawler parameter settings.

	ai		*plants*		*vehicles*	
	nf	5-gram	nf	5-gram	nf	5-gram
downloaded size (raw HTML)	234GB	141GB	292GB	172GB	394GB	168GB
downloaded size (plain text)	21GB	14GB	13.5GB	15.3GB	12.5GB	15.1GB
# documents	36M	17M	3.6M	2.2M	4.1M	2M
# sentences	98M	176M	209M	117M	119M	193M
# relevant sentences	35M	26M	16M	37M	6.6M	3.9M
# unique rel. sent.	24M	14M	13M	16M	5.5M	7.6M

sequence of words into account. Each crawl ran for about five days. On average, about 15 GB of text data, which amounts to roughly 230 GB of raw HTML data and 10 million downloaded documents, were harvested by each crawl. Table 2.4 summarizes some properties of the collected data.

To measure the quality of a crawl during runtime, we define the *harvest rate* at time t to be:

$$\begin{aligned} \textit{harvest rate}(t) &:= \frac{\textit{docs}(t,\epsilon)}{\textit{docs}(t,\infty)} \text{ , where} \\ \textit{docs}(t,\epsilon) &:= \sum_{i=1}^{t} \delta(d_i,\epsilon) \text{ , and} \\ \delta(d,\epsilon) &:= \begin{cases} 1 & \text{if } PP(d) < \epsilon \\ 0 & \text{otherwise ,} \end{cases} \end{aligned} \tag{2.8}$$

In words: after crawling t documents, we measure the ratio of good documents (documents below a threshold ϵ) out of all documents downloaded at time t. In the experiments, the same perplexity threshold value was used to select 'good documents' as was used throughout crawling to prevent links from queuing up. Since perplexity values are subject to a particular language model, and to compare the different runs qualitatively across different models, we computed a perplexity value for each downloaded document regarding a reference 3-gram language model.

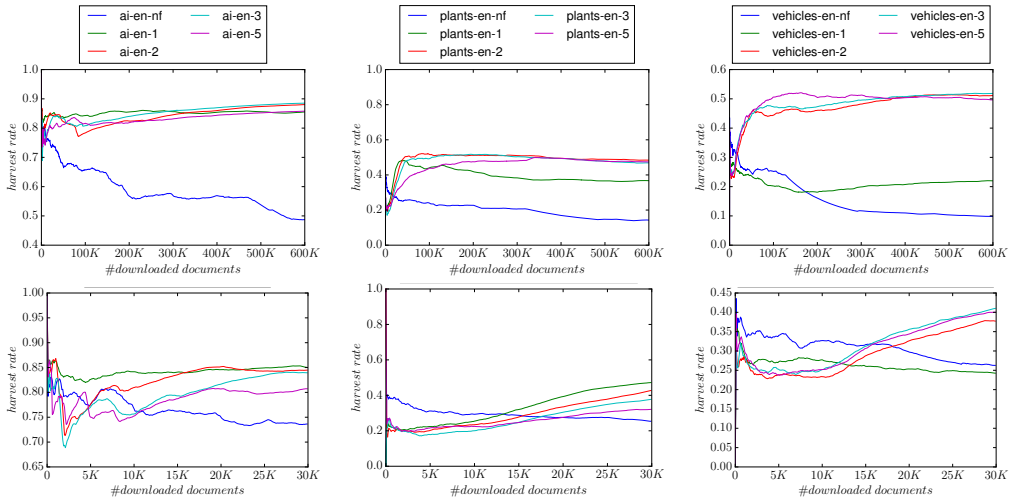


Figure 2.4: Harvest rate vs. time: the ratio of perplexity pruned documents to downloaded documents at time t (top row). The bottom row shows the harvest rate from $t = 1$ to $t = 30{\times}10^4$, i.e., the beginning of each crawl.

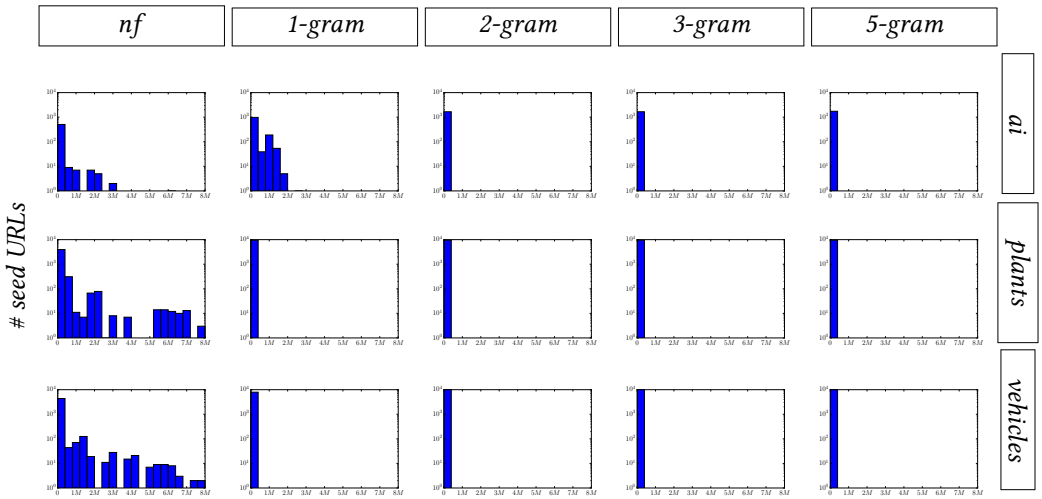


Figure 2.5: Number of downloaded seed URL documents from 0 to 10,000 on a logarithmic scale (y-axis) vs. download time as from first to 8 millionth document.

The results of this study are shown in Figure 2.4, top row. As expected, the different models used for crawling show similar asymptotic behavior for each value of n across the different topics; namely, they have a larger and apparently convergent harvest rate. In the long run, the sequence-based models ($n = \{2, 3, 5\}$) are not clearly distinguishable, i.e., all seem to converge to a higher harvest rate than the bow model ($n = \{1\}$) or the non-focused model. For illustrative purposes, consider *ai*; here, more than 80% of the documents are considered good under a focused crawl with a sequence language model, even after crawling 600K documents, whereas the non-focused crawl 'loses its focus' over time, and the bow models perform not as stable as the sequence models. To summarize, all sequence-based models ($n >= 2$) perform relatively stable across different topics; the bag-of-words model ($n = 1$) performs worse but still better than the non-focused model.

Another interesting observation is that the non-focused crawl has a better or

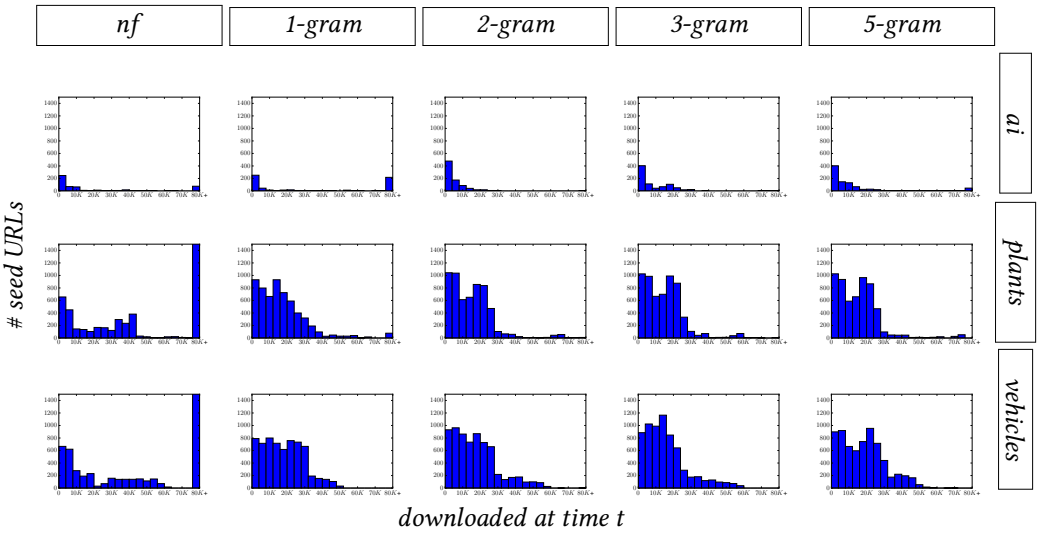


Figure 2.6: Number of downloaded seed URL documents from 0 to 2,000 (y-axis) vs. download time as from the first to the 80 thousandth document. The last bar includes the number of seed URLs downloaded after the 80 thousandth document.

similar harvest rate for the first nearly 15K documents (cf. Figure 2.4, bottom row), whereas the focused crawl needs a 'heating phase' until its harvest rate starts to grow. Simply put, the focused crawls seem to explore uninteresting documents first, whereas the non-focused crawl explores more interesting documents first. As this seems counterintuitive, we believe the only explanation comes from the different strategies for seed processing and also the actual document quality of a seed URL regarding further usability for corpus creation. Note that a good seed URL is a good *hub*, i.e., it contains a lot of URLs to interesting web documents, while it might still be a bad *authority* for our scenario, i.e., it does not contain much valuable text (Kleinberg, 1999).

Since Heritrix's default queuing strategy, mixed with parallelism and politeness settings, is inconclusive to some extent – it is nearly impossible to analyze the timeline exhaustively – we investigated the processing of seed URLs in time and show the results in Figure 2.5 for 8M documents and in Figure 2.6 for the first 80K documents. As becomes apparent, the default (non-focused) strategy downloads seed documents regularly over a long period of time, e.g., for *vehicles* and for *plants*, more than 1,400 seed URLs are downloaded after downloading 80K other documents, in fact, the last few seed URLs were processed after 7.5 million downloaded documents. Note that Heritrix's default crawling strategy is not strictly breadth-first nor depth-first, but a mix of both, which is relatable since its main purpose is to download entire 'snapshots' of the web. However, in the focused crawling strategy, however, seed URLs are initialized with the highest possible priority score to guarantee quick processing. It might still happen, though, that politeness settings limit the access to the seed URL documents; parallel processing of the queue leads to other URLs being crawled first.

Another way of measuring the quality of a downloaded corpus from a language modeling perspective is to create a web-size language model from the downloaded corpora and measure perplexity regarding held-out test documents from the same

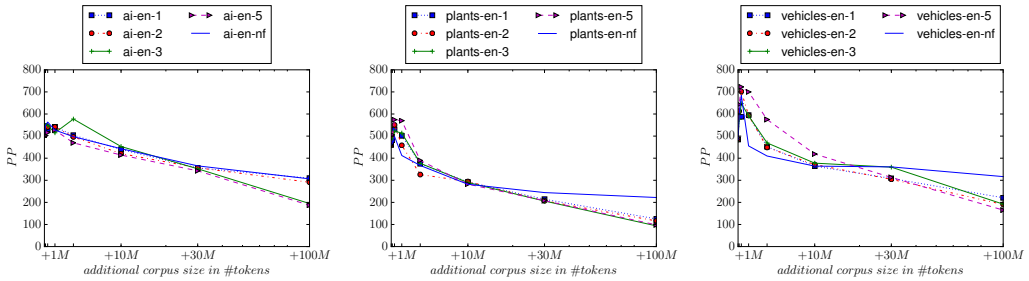


Figure 2.7: Perplexity tested on test documents, with one LM at certain intervals of the crawl.

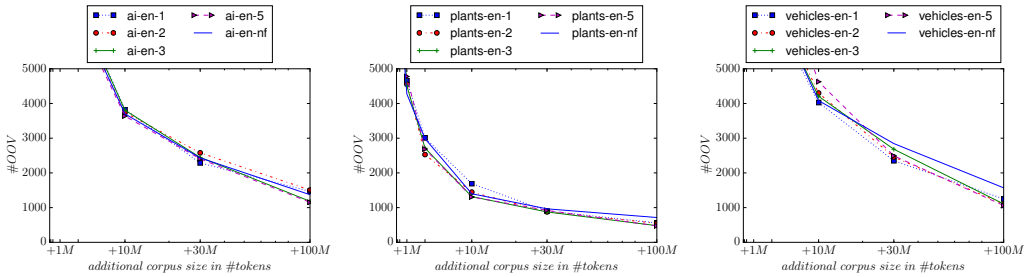


Figure 2.8: Amount of OOVs tested on test documents regarding a certain LM, with one LM at certain intervals of the crawl.

domain – similar to Section 2.5. For this, we again created a language model from each of the crawls after specific amounts of tokens. We added the data to the training corpus to model a corpus enhancement scenario. The initial training corpus size can be seen in Table 2.3, and the resulting perplexity values are plotted in Figure 2.7. The results confirm that the difference in the domain affinity between $n = 2$ and $n = 5$ is marginal, but a substantial difference exists to $n = 1$ and the non-focused setting. We also see the same effect as in the harvest rate evaluation, i.e., for roughly the first 10 to 30 million tokens, approximately around 20K documents, the non-focused crawls perform on par or better than the focused strategies. More importantly, we can also see a decline in the number of out-of-vocabulary words (OOVs) regarding the LMs at each interval (cf. Figure 2.8). The relative differences seem only marginal, though. For further evaluations, we used only those corpora crawled with the 5-gram model since it performs most stable across the domains.

2.7.5 Extrinsic Evaluation: Taxonomy Induction

The TAXI system is then applied using the collected data. To compare the performance of TAXI with a commonly used standard corpus, we included Wikipedia. For a fairer evaluation, we adjusted the size of the crawled data to the size of Wikipedia, which means that we used only the first 11GB of plaintext data crawled in each scenario because our Wikipedia amounts to 11GB. We calculate precision, recall, and F1 measures according to TExEval (Bordea et al., 2015). Table 2.5 to Table 2.7 list the results of the system on the taxonomy induction task. On the

Table 2.5: Results on the taxonomy induction task for *ai*. Relation density is computed as the ratio of unique in-domain relations to corpus size in million.

	wiki	nf	5-grams
# unique relations	8.1M	3.0M	3.1M
# unique in-domain relations	34K	34K	53K
relation density	20.0	25.9	30.2
P	.66	.68	.64
R	.07	.08	.10
F_1	.13	.15	**.18**

Table 2.6: Results on the taxonomy induction task for *plants*. Relation density is computed as the ratio of unique in-domain relations to corpus size in million.

	wiki	nf	5-grams
# unique relations	8.1M	3.6M	4.2M
# unique in-domain relations	52K	55K	112K
relation density	30.1	45.6	63.8
P	.78	.67	.58
R	.54	.45	.56
F_1	.52	.39	**.57**

one hand, most relations are created from Wikipedia; on the other hand, more in-domain relations are created with each focused crawl. We tested in-domain relations by measuring vocabulary overlap with the gold data, i.e., if one part of a relation (hyponym or hypernym) exists in the target taxonomy's vocabulary, we count it as in-domain. The results show that the focused crawls perform best in terms of F_1 score in all domains, whereas a higher recall generally dominates the $F1$ score, which can be attributed to higher coverage. Note that for the domains *ai* and *plants*, the focused crawl attains more unique in-domain relations with fewer unique relations in total than, for example, Wikipedia. Interestingly, this is not the case for the *vehicles* domain. By looking into the data, this can be attributed to the high number of web pages offering cars for sale, which offer fewer taxonomic relations. Also, relation density, i.e., the ratio of in-domain relations vs. corpus size in millions of tokens, is in two out of three domains higher for the focused crawls, which we explain with a much higher domain focus. Since the lower precision for the focused crawls, in general, seems rather counterintuitive, we explain this phenomenon as follows: As the size of the corpus grows, more pruning is needed in order to distinguish good relations from noisy relations since eventually, frequent words tend to be related to everything else. For the focused crawls, domain-specific words are essentially frequent words; thus, we'd need to adjust the pruning parameter in TAXI to balance the number of relations for each corpus.

Table 2.8 shows some correctly and in-correctly extracted examples. The last example for vehicles nicely shows the domain difference to the general domain, i.e., generally spoken, a 'rocket' is considered to be a 'missile', but in the 'vehicles' domain, the term 'rocket' is a short form of 'rocket car', a dragster car powered by a rocket engine and is thus considered to be a vehicle.

Table 2.7: Results on the taxonomy induction task for *vehicles*. Relation density is computed as the ratio of unique in-domain relations to corpus size in million.

	wiki	nf	5-grams
# unique relations	8.1M	2.3M	3.1M
# unique in-domain relations	27K	12K	23K
relation density	15.9	9.7	13.5
P	.70	.64	.50
R	.38	.35	.50
F_1	.49	.45	**.50**

Table 2.8: Examples for extracted relation in the different domains.

Domain	Hyponym	Hypernym	correct
ai	ability	cognitive skill	no
ai	act	event	yes
ai	active learning	instructional method	yes
plants	hellebore	poisonous plant	yes
plants	helianthemum	rockrose	no
plants	redbud	angiospermous tree	yes
vehicles	armored personnel carrier	armored vehicle	yes
vehicles	rocket	vehicle	yes
vehicles	rocket	missile	no

2.8 Conclusion

Due to the dynamic nature of the web, evaluating focused crawling methods is a non-trivial task. We presented a focused crawling approach using statistical N-gram modeling and combined the evaluation with an extrinsic task based on in-domain taxonomy induction. Also, we evaluated the TopiCrawler system in depth regarding its language model parameters and showed that a bow model could be sufficient for guiding a crawl. Still, at least a 2-gram model is recommended. We confirm that it is beneficial to use the sequence capabilities of language models, where the difference in performance is not significantly larger between a 2-gram model and a 5-gram model. The 5-gram model still performs constantly better, though it comes with higher costs for construction, i.e., longer initialization time and slower computation time because of more back-off calculations and more memory consumption. However, since the initial input data is typical of a rather small nature, for example, a couple of hundred documents, those drawbacks are of minor importance. If computational resources are short, though, falling back to a 3-gram or 2-gram model will yield decent results.

We conclude that while the intrinsic performance evaluation suggests that the focused methods have a large margin to the non-focused method (cf. harvest rate in Figure 2.4), the extrinsic evaluation based on taxonomy induction suggests that either *a*) general domain terms still dominate in-domain taxonomies, or *b*) focused web crawls cannot gather the data needed for the in-domain taxonomies. This

suggests that focused crawling should be evaluated on more extrinsic tasks, and only a common improvement will be expressive enough.

We note that the results of the extrinsic evaluation show that the presented approach to focused crawling provides corpora, which yield higher recall than Wikipedia due to higher coverage of in-domain terms. Overall, this research shows that it is possible to automatically extend corpora based on focused crawling with language models. It is beneficial not only for constructing better language models of the targeted domain but also in applications requiring large in-domain collections, such as taxonomy induction. In times where many NLP systems rely on large background corpora for, e.g., computing word embeddings or N-gram language modeling, focused crawling is a viable and straightforward way to grow one's relevant background text collection. Note that even more precise in-domain corpora can be compiled by further pruning the downloaded documents using the same ranking methodology as during crawl time, i.e., the same language model and perplexity scoring.

All software packages are made available as open-source applications under permissive licenses.[22,23]

22. https://tudarmstadt-lt.github.io/topicrawler/
23. https://github.com/tudarmstadt-lt/seg

This chapter is based on the following published work by the author and collaborators:

Dirk Goldhahn, **Steffen Remus**, Uwe Quasthoff, and Chris Biemann. 2014. Top-Level Domain Crawling for Producing Comprehensive Monolingual Corpora from the Web. In *Proceedings of the LREC-14 workshop on Challenges in the Management of Large Corpora (CMLC-2),* 10–14. Reykjavik, Iceland.

Alexander Panchenko, Stefano Faralli, Eugen Ruppert, **Steffen Remus**, Hubert Naets, Cedrick Fairon, Simone P Ponzetto, and Chris Biemann. 2016. TAXI at SemEval-2016 Task 13: A Taxonomy Induction Method based on Lexico-Syntactic Patterns, Substrings and Focused Crawling. In *Proceedings of the 10th International Workshop on Semantic Evaluation,* 1320–1327. San Diego, CA, USA.

Steffen Remus. 2014. Unsupervised Relation Extraction of In-Domain Data from Focused Crawls. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics,* 11–20. Gothenburg, Sweden.

Steffen Remus and Chris Biemann. 2016. Domain-Specific Corpus Expansion with Focused Webcrawling. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016),* 23–28. Portorož, Slovenia.

Steffen Remus, Gerold Hintz, Darina Benikova, Thomas Arnold, Judith Eckle-Kohler, Christian M Meyer, Margot Mieskes, and Chris Biemann. 2016. EmpiriST: AIPHES Robust Tokenization and POS-Tagging for Different Genres. In *Proceedings of the 10th Web as Corpus Workshop (WAC-X),* 106–114. Berlin, Germany.

3
Sense Induction

In this chapter, we discuss semantic structuring on various levels, e.g., grouping semantically related words into clusters is one form of semantic structuring, where the result could also be interpreted as topics. We present ideas of measuring semantic similarity (or semantic relatedness) between words and use similarity graphs to cluster those words into so-called sense clusters.

Word embeddings – generated with neural networks (NN) or other matrix factorization techniques – are an important tool in natural language processing (NLP). *Embeddings* are lower dimensional dense representations of a larger, potentially very sparse vector space. Unless stated otherwise, we refer to *word embeddings*, i.e., dense vector space representations of words, tokens, or word-pieces (Baeza-Yates and Ribeiro-Neto, 1999), when we mention the more general term embeddings in this work. Standard *vector space models* (VSMs) of words already provide intrinsic semantic properties, just because its original dimensions, or more generally, the features of a word during the optimization process, depending on the context of the occurrence of a word (Mikolov, Yih, et al., 2013), thus the *Distributional Hypothesis* (DH; Harris, 1954) is implicitly applied by design.

In this work, we focus mainly on embeddings generated by NNs – so-called neural word embeddings – because of their superior performance and ongoing impact in NLP research. However, we note that our findings apply to other types of embedded word vector spaces too – we present experiments with similar outcomes using LSA vector representations.

3.1 Sense Induction by Retrofitting Static Word Embeddings

An essential issue with static word embeddings (SWEs) is the lack of sense-awarenes, i.e., a word and its vector share a bijective mapping and ignore the multiplicity of meanings a word can bear, e.g., the word 'iron' may refer to an atomic element, a device for smoothing clothes, a golf club, a color, and, due to the changing nature of language, potentially infinitely many more. Word-

NET (Fellbaum, 1998) defines four different interpretations of 'iron'. All of those different meanings, however, share the same point in the vector space, and still, these vectors are used in downstream tasks such as sentiment analysis, named entity recognition, question answering and many others, which might lead to misinterpretations and error propagation.

In this work, we will empirically show that word embeddings mainly keep dominant senses in the vicinity of an ambiguous word, as defined by the background corpus they were estimated from; and we illustrate this issue based on multiple examples. Sense inventories are required to distinguish different word senses, and we argue that meanings, rather than words, should be represented in the vector space. We strive for several distinguished vectors per ambiguous word, corresponding to its senses (Navigli, 2009; Denkowski and Lavie, 2014). In addition, we show that embeddings can hardly be used for unsupervised *word sense induction* (WSI), i.e., to create such sense inventories automatically. Accordingly, we use external resources as sense-inventory (manually created and automatically generated), showing substantial relative improvement for multiple word-similarity tasks. Also, we believe senses cannot be 'hard-coded', i.e., the number of senses and their distinctions depend on time and domain, which is why we argue for inducing word senses in an unsupervised fashion, i.e., clustering in a post-processing sense-induction scenario, see, e.g., (Navigli, 2009; Denkowski and Lavie, 2014) for further details.

Hypothesis: In (**Remus** and Biemann, 2018), we hypothesize that retrofitting existing word embeddings to gain sense-aware word embeddings, or so-called sense vectors, is beneficial, even for word similarity computations. For this, we use word sense induction techniques and apply the resulting sense vector representations to the word-similarity task. Particularly for rare word combinations, we expect that minor senses will have a more balanced effect, whereas without retrofitting, major senses dominate in word-similarity computations. We verify our hypothesis for multiple embeddings from four monolingual corpora and one corpus computed from paraphrases. We present relative improvements on various word similarity benchmark datasets and show that minor senses are indeed the reason for improvements. Additionally, we show two alternative baselines: AUTOEXTEND (Rothe and Schütze, 2015) and ADAGRAM (Bartunov et al., 2016), which model senses as part of the training process for so-called neural sense embeddings. To the best of our knowledge, we were the first to employ word sense induction techniques for retrofitting single word vectors to the multiplicity of their meanings, creating new word-sense vectors, and using those for semantic similarity by pro-actively addressing minor senses (**Remus** and Biemann, 2018).

3.1.1 Related Work

Although word similarities are rarely used explicitly in downstream tasks, it goes without saying that semantic properties such as semantic similarity are implicitly necessary. A number of word similarity benchmarks exist for the purpose of intrinsically testing semantic properties of word embeddings (Hill et al., 2014; Finkelstein et al., 2001; Bruni et al., 2014; Gerz et al., 2016). Word similarities are

usually measured with cosine similarity of two given vectors $\mathbf{u}$ and $\mathbf{v}$

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}^\top \mathbf{v}}{||\mathbf{u}|| \times ||\mathbf{v}||} \,. \tag{3.1}$$

$\mathbf{u}$ and $\mathbf{v}$ can be represented by words or senses; we will outline further details in the next subsections.

Sense Embeddings

Integrating sense information in embeddings is still an active line of research. For reference, we compare our results to existing projection models, which include sense induction or sense labeling in their learning process. Rothe and Schütze (2015), for example, introduced AutoExtend, a neural network model which enriches existing embeddings with word sense information by explicitly learning the representations.[1] The sense inventory is taken from WordNet but Rothe and Schütze (2015) particularly emphasize that any lexical or semantic resource could be used. We tested their provided embeddings, which extend Mikolov, Chen, et al. (2013)'s sgns model and used them for comparison. Neelakantan et al. (2014) and Bartunov et al. (2016) present neural architectures which gather sense information purely from monolingual text. We compare to AdaGram (Bartunov et al., 2016) as an additional baseline because it compares favorably to (Neelakantan et al., 2014). AdaGram's main parameter essentially regulates the maximum number of senses per word, but the algorithm finds the number of senses automatically in this range. Eventually, each word has one or more corresponding sense vector. We report the results of this method in Section 3.1.4.

Retrofitting

Retrofitting is the process of editing or fitting a given item for a specific task. Faruqui et al. (2015) defines retrofitting as a post-processing objective that improves existing word embeddings. Multiple objectives have been defined, e.g., Faruqui et al. (2015) and Kiela et al. (2015) use lexical resources for the retrofitting objective, while, for instance, Wieting et al. (2015) uses a database of aligned paraphrases.

3.1.2 Methodology

In the remainder of this section, we will use v to refer to a word and $\mathbf{v}$ to refer to v's corresponding word vector.

Unsupervised Sense Inventory

Our proposed method solely relies on a word vector space and an appropriate sense inventory. We define a synset for a word v to be the set of similar words that express the same meaning, i.e., one shared meaning, and the sense inventory of v to be the collection of its synsets, i.e., the different senses v can bear. We follow Dorow and Widdows (2003), Pantel and Lin (2002), and Pelevina et al. (2016), and use an unsupervised WSI method, which provides us with so-called *unsupervised*

1. http://www.cis.lmu.de/~sascha/AutoExtend/

synsets. The simplified procedure to compute an unsupervised synset given the word v is the following:

1. compute the **top n nearest neighbors**
2. compute a similarity score between every pairwise combination of neighboring words which renders a **similarity matrix** or a so-called fully connected *similarity graph*
3. compute a **clustering**, where ideally, each cluster of words represents a different sense of v.

This general methodology has been proven to perform sufficiently well on a number of NLP tasks, although details of the procedure might vary. We test various parameters in each step of this methodology, which we explain in more detail in the next sections.

Similarity Matrix Computing a similarity matrix M between terms u and v from the respective embedding matrix is straightforward: we use the cosine similarity such that the entry $M_{uv} = M_{vu} = \cos(u, v)$. Computing a similarity matrix M for symbolic/graph-based thesauri is done by retrieving the top 50 similar terms for every word w_i and setting $M_{ij} = 1$ for every word $w_j \neq w_i$ iff w_j is contained in w_i's list of top 50 similar terms or vice versa:

$$M_{ij} = M_{ji} = 1 \text{ iff } w_j \in \text{top}_{50}(w_i) \vee w_i \in \text{top}_{50}(w_j) \; . \tag{3.2}$$

Clustering Since words cannot be expected to have a fixed number of senses, we tested two graph-based clustering algorithms, where the algorithm itself determines the number of clusters. Because of its symmetry, the similarity matrix can be interpreted as an *adjacency matrix* for an *undirected similarity graph*. We tested the following graph clustering algorithms: 1.) CW (Chinese Whispers; Biemann, 2006), 2.) MCL (Markov Clustering; van Dongen, 2000).[2] Graph clustering algorithms perform best if the adjacency matrix is sparse. To make the fully connected dense similarity matrix sparse, we prune M by a threshold parameter τ_{cos}:

$$M_{ij} = 0 \text{ iff } M_{ij} < \tau_{cos} \; . \tag{3.3}$$

Apart from that, we use the default parameter settings suggested by Biemann (CW; 2006) and van Dongen (MCL; 2000). Singleton clusters, i.e., clusters which contain only one element – which occur frequently for large τ_{cos} – are merged into one 'residual' cluster. The resulting clustering represents the collection of synsets S_v of the word v; we refer to a particular synset or sense k of v as S_v^k. We want to stress that v is not contained in any of its synsets, i.e., $S_v^k = S_v^k \setminus v$ per definition.

Additionally, we use an unsupervised sense inventory, pre-computed by Biemann and Riedl (2013) and Riedl (2016) using the JoBimText (JBT) framework,

2. We also tested other clustering algorithms, such as k-means and self-organizing-maps, for comparison purposes but report results only for CW and MCL since they do not depend on the number of clusters as a parameter and yield visually better clusters.

which produces a graph-based sparse model.[3] JBT also applies the CW algorithm for inducing word senses based on counting context overlaps, but JBT allows to do this on a large scale due to its implementation in a multi-computer setup using the hadoop ecosystem[4]. This resource provides a different, more diverse set of neighboring words than the vector space neighborhoods, as we shall examine in Section 3.1.4.

Retrofitting Word Embeddings

The main idea of retrofitting word vectors to sense vectors is to find a unique representation for a particular sense of a word. Using sense inventories, the individual word vectors from a particular synset, which describe a unique sense of the target word, are merged such that a different vector will represent each sense of a word (resp. synset). For a target word v, we average all vectors $\mathbf{u}$ of words u in the synset S_v^k and add the vector $\mathbf{v}$ with weight λ in order to compensate for semantic drift, for which we found strong indications in preliminary experiments:

$$\mathbf{v}_k = \lambda \mathbf{v} + (1 - \lambda) \sum_{u \in S_v^k} \frac{\mathbf{u}}{|S_v^k|} \,, \tag{3.4}$$

where λ is a scalar weighting factor in $[0, 1]$. A geometric interpretation of this is first to find a cluster's center and then shift the center by λ into the direction of the target word. Note that the clustering for any word v is performed without v itself, i.e., it is not contained in the sense inventory, cf. (Dorow and Widdows, 2003), hence the shifting.

We experimented with the size of S_v^k. This stems mainly from the observation that clusters naturally have different sizes during clustering. The largest clusters often refer to major senses, whereas smaller clusters are usually minor senses – as defined by the background corpus from which the model was estimated. To alleviate the effect of averaging noisy words in large clusters, we select only the top m words in a synset, as defined by the clustering.

Sense-Aware Word Similarities

We tested different procedures for computing sense-aware similarities between any two words u and v:

$$\text{sim}(u, v) = \arg\max_k \; \cos(\mathbf{u}_k, \mathbf{v}) \tag{3.5}$$

$$\text{sim}(u, v) = \arg\max_l \; \cos(\mathbf{u}, \mathbf{v}_l) \tag{3.6}$$

$$\text{sim}(u, v) = \arg\max_{k,\, l} \; \cos(\mathbf{u}_k, \mathbf{v}_l) \,. \tag{3.7}$$

Equations (3.5-3.7) involve finding the nearest senses k and l for the words u and v. We compare these metrics to the standard cosine similarity $\cos(\mathbf{u}, \mathbf{v})$, which is not sense-aware.

3. see http://ltmaggie.informatik.uni-hamburg.de/jobimviz (Ruppert et al., 2015)
4. https://hadoop.apache.org/

3.1.3 Experimental Setup

Word-similarity Benchmark Datasets

Hill et al. (2014) define that there must be strong distinction between similarity and relatedness between word pairs. While related words roughly fit into the same topic, similar words are more specific; they fit into the same topic and constitute (partial) substitutability. Consider, for example, the words *student* and *professor*, which are undoubtedly related but not similar because they share only a few contexts in which the two words can be exchanged unconditionally, hence they are considered highly dissimilar due their antonymic nature, while *lecturer* and *professor* might be exchangeable in more context depending on the situation and are thus related and more similar.

The WordSim353 (Finkelstein et al., 2001) dataset describes relatedness values of 353 noun pairs, and the SimLex999 dataset represents word similarity for 666 noun pairs, 222 verb pairs and 111 adjective pairs. Particularly the latter's emphasis is to model opposite meanings (antonym-like) as very non-similar.

Another dataset is the MEN[5] dataset (Bruni et al., 2014), which models, analog to WordSim353, relatedness or association rather than similarity. Bruni et al. (2014) randomly sampled 3, 000 word pairs from words that occur at least 700 times in the ukWaC + Wackypedia combined corpora.[6] MEN comprises of inter part-of-speech word pairs, e.g., pairs like (*apple*, noun : *orange*, adjective) or (*bear*, verb : *boxer*, noun). It is also worth noting that MEN comes in two forms, *a*) in a lemmatized form with POS-tags, and *b*) in natural surface form. We report results on the former, lemma form with POS-tag information.

The verbsim dataset (Gerz et al., 2016) can be interpreted as a larger version of the verb part of SimLex999, exclusively containing 3, 500 verb pairs, allowing more meaningful benchmarking with more and better-represented examples due the use of external resources such as VerbNet.

Embedding Matrices

We used pre-trained word vectors provided by Mikolov, Chen, et al. (2013), which were trained on Google News texts containing 6 Billion words.[7] Using those pre-trained SGNS embedded vectors and cosine similarity between words yields a baseline Spearman correlation score of $\rho = 0.44$ on the SimLex999 dataset. Additionally, we use GloVe[8] (global vectors; Pennington et al., 2014) embeddings, which yield a baseline correlation score of $\rho = 0.37$ for SimLex999.

Schwartz et al. (2015) defined the context of a word to be the *symmetric pattern* it occurs with. A symmetric pattern is a shallow pattern in the form of *'X or Y'*, *'X and Y'*, *'X as well as Y'*, *'X rather than Y'*, where particular instances of *X* and *Y* occur in both positions, e.g., *'cats and dogs'* and *'dogs and cats'* are considered instances of a symmetric pattern, while, e.g., *'point of view'* cannot be altered without losing

5. https://staff.fnwi.uva.nl/e.bruni/MEN

6. http://wacky.sslmit.unibo.it/

7. We used the 300-dimensional model trained on Google News. The model and the source code are available at https://code.google.com/p/word2vec/.

8. We use the 6 Billion word, 300-dimensional model available at http://nlp.stanford.edu/projects/glove/.

its meaning, the pattern *'X of Y'* is thus asymmetric. Some symmetric patterns are considered to be particularly indicative for antonymy, e.g., *'either X or Y'* or *'rather X than Y'*. Schwartz et al. (2015) used symmetric patterns to build an antonym-sensitive embedding model from monolingual corpora. We use their 10K dimension model built on an 8G words corpus[9] and refer to this embedding type as SYMPAT. We tested the 300 and 500-dimensional vectors provided by Schwartz et al. (2015), but the 10K version achieved the best results among the SYMPAT embeddings. All SYMPAT reported results are thus based on the 10K dimensional embedding.

Wieting et al. (2015) used PPDB (paraphrase database; Ganitkevitch et al., 2013) pairs to train a projection matrix called PARAGRAM. The matrices are initialized with the GLOVE embeddings and fitted to match with PPDB's paraphrases. By using PPDB, the model is already guided to represent synonymous expressions with similar vectors, as opposed to expressions with opposite meanings. Wieting et al. (2015) further tuned the matrices, e.g., they performed hyperparameter optimizations on WORDSIM353, resulting in PARAGRAMWS, and SIMLEX999, resulting in PARAGRAMSL. These embeddings are thus tuned for **either** relatedness **or** similarity and build a strong baseline with a correlation score of $\rho = 0.68$ on SIMLEX999.

Wieting et al. (2016) later introduced CHARAGRAM embeddings which further improved on the SIMLEX999 task by using character *n*-grams so that the model can also account for unknown words that are not in the training corpus. Noteworthy is also the work by Recski et al. (2016), who further improved the SIMLEX999 results by using concept networks and strong supervision, and Mrkšić et al. (2016) who used counter-fitting, a synonymy versus antonymy injection method, and improved results up to $\rho = 0.77$ and $\rho = 0.74$ respectively. However, we deliberately do not go into details here since these supervised models are out of the scope of this work: we focus on the relative improvement of monolingual embeddings by exploiting unsupervised WSI methods and stay agnostic as to whether antonyms are similar or dissimilar since they are mostly similar except in one semantic dimension. We are thus independent of any manually developed resource.

We also make use of two LSA(latent semantic analysis) embeddings trained on English corpora and provided by Günther et al. (2015).[10] Both models are based on a 2-Billion-word corpus and use a *positive pointwise mutual information* weighting scheme (PPMI) before applying *singular value decomposition* (SVD) with 300 target dimensions and a vocabulary of 100K words. We refer to the model based on a bag-of-word representation of documents as LSABOW and to the model applying a HAL-like (Hyperspace Analog to Memory) context representation as LSAHAL (following terminology of Günther et al., 2015). HAL is based on a 10-word moving word window, where words that are used in related contexts (within the same window) have high similarity (Burgess, 1998).

3.1.4 Results

Word Sense Induction

We compute the top 500 neighboring terms from a word vector by means of cosine similarity. Clearly, the number of nearest neighbors defines the vocabulary of the

9. http://homes.cs.washington.edu/~roysch/papers/sp_embeddings/sp_embeddings.html

10. Models are available for download under http://www.lingexp.uni-tuebingen.de/z2/LSAspaces/.

Table 3.1: Examples of related words according to the respective technique. Superscript numbers refer to senses, * is another syntactic form of the query word.

word	SGNS	JBT
iron	irons*, wood[1], wedge[1], fairway[1], niblick[1], putter[1], mashie[1], tee[1], bunker[1], steel[2], …	steel[2], metal[2], aluminum[2], copper[2], zinc[3], calcium[3], putter[1], wedge[1], sand[1], …
mob	Mob*, gangsters[1], mobs*, mafia[2], mobsters[2], mobster[2], gangster[1], …	gang[1], crowd[3], protester[4], demonstrator[4], Mafia[2], thug[1], horde[3], rioter[4], militia[4], gunman[1], …
class	classes*, classs*, Class*, grade[1], middle[2], precalculus[1], semesterlong[1], grades[1], studens[1], …	lesson[4], Class*, workshop[2], instructor[3], seminar[2], training[4], course[2], session[4], classroom[5], lecture[2], autumn[5], test[6], exercise[6], department[5], college[5], …

Table 3.2: Average number of clusters for varying $\tau_{\cos}$ and the two graph clustering algorithms for three embeddings.

		τ_{cos}						
		0.3	0.4	0.5	0.6	0.7	0.8	0.9
MCL	SGNS	1.7	13.5	34.5	50.4	38.5	10.2	2.4
	paragramWS	11.0	33.7	60.3	78.0	78.4	59.1	23.1
	LSAbow	1.1	1.4	2.6	8.0	24.0	47.7	44.3
CW	SGNS	1.4	3.0	7.9	16.2	15.0	4.0	1.5
	paragramWS	2.5	5.3	12.3	25.4	33.5	25.3	8.0
	LSAbow	1.1	1.2	1.6	2.7	5.9	13.0	14.1

sense inventory of a word. We can confirm the observation of Faruqui et al. (2016) and Schnabel et al. (2015) that within neural word embeddings, the frequency rank of a word's neighbor depends on the frequency rank of the word itself. This is clearly an issue because the frequency of a word's sense also correlates with the frequency of a word's occurrence. It makes local sense clustering challenging because words indicating different senses are not found in a word's vicinity but stretch across the entire vocabulary space. Cosine similarity can account for that, words referring to different senses will point to different directions, but in order to do local sense clustering, one would need to cluster the entire vocabulary, which is computationally too expensive to be considered in practice. On the other hand, it is well-known that graph-based methods can hardly be used to compute word similarities between arbitrary words because of the nature of its computation, i.e., there is often no context overlap between words after the graph pruning step (Riedl, 2016).

Table 3.3: Cosine similarity (cos) and similarity by the number of shared contexts (#ctx), next to the relative rank regarding cos for SNGS and #ctx for JBT with respect to the query word '*iron*'.

	SGNS		JBT		
related term	cos	rank	#ctx	rank	sense label
putter	0.46	17	36	128	golf
wood	0.47	11	119	15	sports
copper	0.37	252	206	9	metallic
aluminum	0.35	427	206	8	elements
salt	0.23	23,731	31	158	nutrition
fiber	0.20	47,072	77	38	
steam	0.12	416,270	28	181	smoothing
shirt	0.12	415,080	–	–	clothes

Table 3.4: Spearman correlation scores on the different datasets and embeddings. Sense-aware similarities are marked with '-S'. The best-performing method is underlined or marked in bold. We distinguish underlined values to be the winning system with a slight margin (< 0.03) and boldface values with a larger margin. We marked paragramSL and paragramWS for SimLex999 and WordSim353 in gray since the method's hyperparameters were optimized on the respective dataset, thus, the results are not comparable. The lower part evaluates only the noun pair parts of the datasets.

	AUTOEXTEND	*ADAGRAM*	*SGNS*	*JBT-SGNS-S*	*GLOVE*	*JBT-GLOVE-S*	*SYMPAT*	*JBT-SYMPAT-S*	*LSA$_{BOW}$*	*JBT-LSA$_{BOW}$-S*	*LSA$_{HAL}$*	*JBT-LSA$_{HAL}$-S*	*PARAGRAMSL*	*JBT-PARAGRAMSL-S*	*PARAGRAMWS*	*JBT-PARAGRAMWS-S*
SimLex999	0.45	0.29	0.44	0.46	0.37	**0.41**	0.54	0.55	0.30	**0.39**	0.27	**0.38**	0.68	0.64	0.66	0.64
MEN	0.72	0.67	0.77	0.78	0.73	**0.77**	0.53	**0.68**	0.67	**0.70**	0.71	**0.74**	0.77	**0.80**	0.80	0.81
SIMVERB	0.43	0.27	0.36	**0.39**	0.23	**0.30**	0.37	**0.45**	0.15	**0.22**	0.19	**0.28**	0.53	0.53	0.51	0.50
WordSim353	0.58	0.61	0.70	0.69	0.61	**0.65**	0.47	**0.62**	0.67	0.66	0.59	**0.63**	0.72	0.73	0.77	0.75
SimLex999-N	0.44	0.33	0.45	**0.50**	0.39	**0.47**	0.48	**0.55**	0.32	**0.46**	0.34	**0.44**	0.68	0.66	0.64	0.64
MEN-N	0.72	0.68	0.77	0.79	0.76	**0.80**	0.57	**0.74**	0.71	0.73	0.73	**0.76**	0.78	**0.81**	0.80	0.82

Recap, to identify senses, the first step is to generate a vocabulary to be divided into synsets, or sense descriptions. Since the vocabulary for these sense descriptions is considered to be related to the original word, it is generated by searching in the neighborhood of a word in the vector space. In the case of embeddings (neural or factorized), however, the immediate neighborhood of a word consists mainly of one dominating sense.

Table 3.1 illustrates this issue by showing neighboring/related words for a

few selected polysemous example words ('iron', 'mob', 'class'). For conciseness and comparison purposes, we only present neighboring words according to SGNS embedding similarity and JBT. For illustration, we manually labeled the top 40 words and tried to pick expressive examples with different senses. As can be seen from the example, SGNS neighbors cover fewer senses than JBT. This is clearly related to the frequency issue described in more detail by Schnabel et al. (2015).

For clustering senses from the embeddings, we tested different values of τ_{cos} (Eq. 3.3). Table 3.2 presents an overview of the results, where we show the average number of obtained clusters/senses. For brevity, we only show results for one of each embedding type, i.e., SGNS as an input embedding, PARAGRAMWS as a retrofitted input embedding, and LSABOW as a non-neural word embedding. Those results highlight the sensitivity of the clustering methods regarding input the type of input embedding. E.g., while τ_{cos} changes from 0.6 to 0.7 MCL delivers roughly -12 clusters for SGNS, almost the same number of clusters for PARAGRAMWS and $+16$ clusters for LSABOW. Based on this initial experiment, we fix $\tau_{cos} = 0.5$ and use CW because this configuration gives us the most stable clusterings across different embeddings, with an average number of clusters being around 7. For the JBT resource, we use the pre-computed sense clusters, which have been computed by using CW too, and that have 3.73 clusters on average.

Initial experiments revealed that directly clustering word embeddings using their vector similarity is not beneficial. We observed a decline in Spearman correlation scores, and data analysis reveals that mostly only one sense per word is obtained. For illustration purposes, consider the example given in Table 3.3, where we highlight scores and ranks for the polysemous word 'iron' to some selected words representing different senses of 'iron'. Mainly terms referring to a 'golf sports' related sense can be found in the immediate vicinity of 'iron', while other terms referring to common senses are further, equally far, away from 'iron'. We have observed this effect consistently for many terms. Because of those shortcomings, we focus our further analysis on utilizing existing JBT clusterings since they have proven helpful for sense induction in the past, e.g., in (Panchenko, Ruppert, Faralli, et al., 2017).

Sense-Aware Word Similarities

Throughout the evaluation, we use the Spearman rank-correlation coefficient ρ. As one of the baselines, we also used all top n neighbors of a word and treated it as a clustering with one synthetic cluster when computing sense-aware similarities as described in Section 3.1.2. We evaluated all datasets for all methods but restrict our discussion to the most interesting results. Selecting the top 5 cluster words proved most useful; in our experiments, we found fluctuating best-performing values between top 3 and top 10, with 5 being among the best values. Also, Equation (3.7) distinguished itself as the best-performing method with $\lambda = 0.5$. Other similarity computations, Equations (3.5;3.6), perform non-satisfactory, sometimes even with a decline in performance. In the remainder of this work, we refer to embeddings with the suffix -S to the sense-aware similarities, which performed best in our previous experiments using the fixed parameters $m = 5$ and $\lambda = 0.5$.

We report AUTOEXTEND (Rothe and Schütze, 2015) and ADAGRAM (Bartunov et al., 2016) scores for comparison. Table 3.4 shows the final results using sense-

unaware similarities, i.e., standard cosine similarity, and our new sense-aware similarities based on the JBT sense inventory.

The results clearly show that sense-aware similarities perform consistently better than their sense-unaware counterparts, e.g., the improvement for the sense-unaware SYMPAT to the sense-aware SYMPAT-S is impressive 0.15. The difference between most sense-unaware systems to their sense-aware counterpart is between 0.02 and 0.15. Particularly, previously inferior embeddings regarding the similarity or relatedness task, e.g., GLOVE or both LSA embeddings, gain most and more consistent from this representation. The loss of performance with the PARAGRAM* embeddings is mainly due to the fact that they have already been optimized for synonymy and antonymy. Injecting the JBT sense inventory – which has no special treatment for antonyms – attracts antonymous or unrelated words again. In fact, this happens to a large extent on adjectives, causing the largest losses. We see consistent improvements across all datasets when looking at the performance for nouns (lower part of Table 3.4).

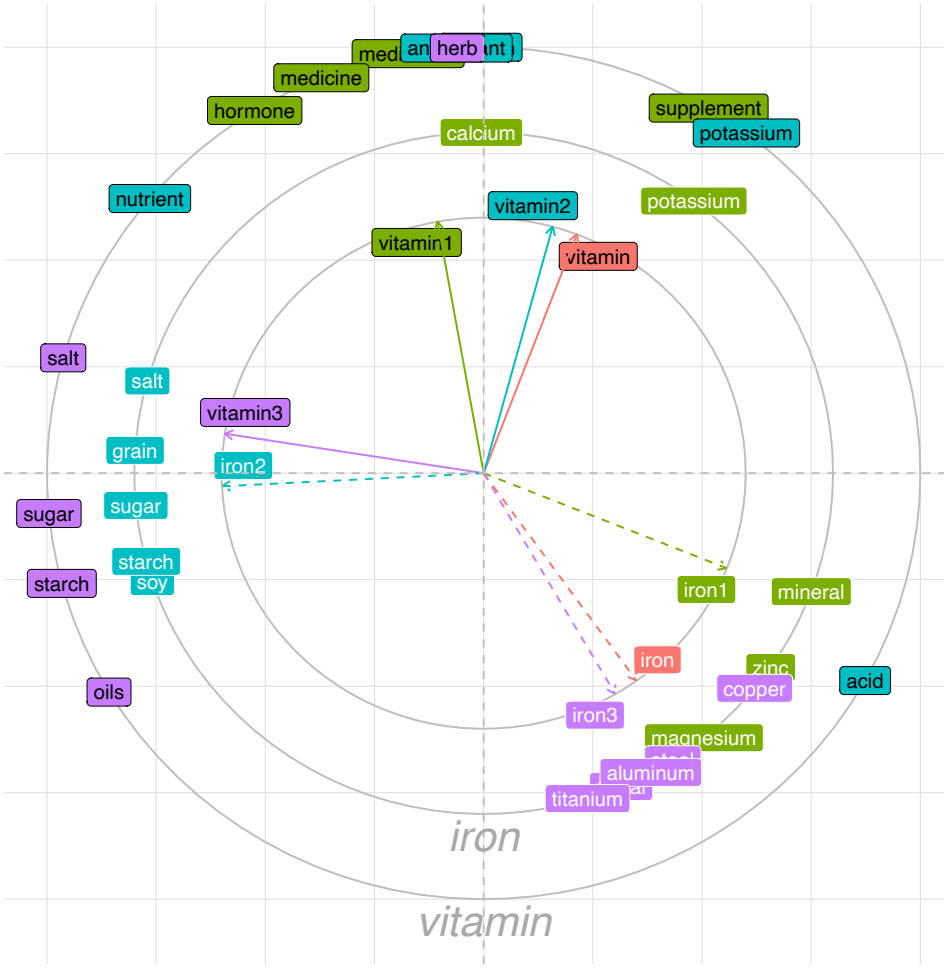
We observe minor sense selections in 3,953 out of 7,734 examples across all datasets for SGNS-S, that is ∼52%. Summarizing, a minor sense was selected in about half of the labeled word pairs. This is most consistent across nouns and varies for verbs and adjectives, which could be attributed to coverage issues[11], or inadequate clusterings for adjectives and verbs since the JBT sense clustering mainly focusses on nouns.

For illustration of adequacy, consider the word pair (‘iron’ : ‘vitamin’) from the SIMLEX999 dataset. Figure 3.1 provides details for the example word pair, including scores and the induced sense inventory. We can see that the SIMLEX999 score is in the mid-range (5.55 out of 10), standard cosine similarity ranks[12] this example at position 212 with a similarity score of 0.22, which is rather low. The sense-aware similarity score selects a link between two suitable minor senses. The particular sense inventory with the top 5 terms can be found in Table 3.1 too. In the example, averaging the sense terms yields a sore of 0.88, which results in a far too high rank of 763, but shifting seems to exploit the perfect balance. The visualization shows the two words and their cluster terms, as well as the averaged cluster centers on the unit circle. Projection was done with T-SNE (Maaten and Hinton, 2008). For better illustration, we mapped cluster terms for each word on a different circle, but note that each circle preserves directions and represents a scaled unit circle. In this visualization, it is easily recognizable that the vectors for ‘iron’ and ‘vitamin’ are still quite far apart, whereas, for example, the retrofitted vectors *iron2* and *vitamin3* are nearby in terms of their cosine similarity.

We computed cross-correlation scores between the methods, e.g., the Spearman correlation score between SGNS and SGNS-S embeddings yield $\rho = 0.85$. This suggests that the individual scores differ, although final SIMLEX999 correlation scores do not seem to benefit drastically (e.g., +0.02 difference for SGNS to SGNS-S).

11. Coverage is around 98% for SYMPAT and 99% for others.
12. Note that Spearman correlation compares ranks.

SimLex999 / rg(SimLex999)	5.55 / 622
SGNS / rg(SGNS)	0.22 / 212
SGNS-S / rg(SGNS-S)	0.59 / 488
$\lvert S^k_{\texttt{iron}}\rvert$	{88, **5**, 98}
$\lvert S^k_{\texttt{vitamin}}\rvert$	{59, 88, **53**}



$S_{\texttt{iron}} = \{\{\texttt{zinc}, \texttt{calcium}, \texttt{magnesium}, \texttt{potassium}, \texttt{mineral}\}, \underline{\{\texttt{sugar}, \texttt{salt}, \texttt{grain}, \texttt{soy}, \texttt{starch}\}}, \underline{\{\texttt{steel}, \texttt{metal}, \texttt{copper}, \texttt{aluminum}, \texttt{titanium}\}}\}$

$S_{\texttt{vitamin}} = \{\{\texttt{supplement}, \texttt{hormone}, \texttt{pill}, \texttt{medication}, \texttt{medicine}\}, \{\texttt{nutrient}, \texttt{calcium}, \texttt{antioxidant}, \texttt{acid}, \texttt{potassium}\}, \underline{\{\texttt{sugar}, \texttt{salt}, \texttt{oils}, \texttt{starch}, \texttt{herb}\}}\}$

Figure 3.1: Scores achieved by sense-aware and sense-unaware word similarity computation for the word pair ('iron' : 'vitamin'). rg(·) refers to the rank regarding the SimLex999 dataset. Selected clusters by the method are written in boldface or underlined. Visualization is based on terms on the unit circle. Every circle represents a unit circle in a joined plot for illustration purposes. The inner circle shows the different sense vectors and the original vectors, the middle circle shows clustered terms generated by the word 'iron', and the outer circle represents clustered words generated by the term 'vitamin'.

To quantify the minor sense effect, we took the SimLex999 dataset and computed an interestingness score for each word pair. The interestingness is defined as a combination of rank differences:

$$\begin{aligned} \text{interestingness} = (\ & \max(\text{rg}(\text{SimLex999})) \\ & - |\,\text{rg}(\text{SimLex999}) - \text{rg}(\text{SGNS-S}|\ ) \\ & + |\,\text{rg}(\text{SGNS-S} - \text{rg}(\text{SGNS})| \end{aligned} \tag{3.8}$$

The intuition of this score is to maximize the difference of the baseline system (SGNS) compared to the improved system (SGNS-S) while minimizing the difference of the human rating (SimLex999) to SGNS-S. Sorting the list of examples in descending order by interestingness provides an overview of which examples were predicted wrong by the baseline system but correct by our system. Sorting the list in ascending order gives us examples of both systems performing equally bad compared to the human rating. The interestingness score of the ('`iron`' : '`vitamin`') example is 1040. We use this scoring and quantify the selection of minor senses in Figure 3.2.

As expected, we found that minor senses are most beneficial for ambiguous examples, which were mispredicted before, but at the same time, the method introduces more noise for words that are rather unambiguous or antonymy has already been modeled. Minor senses are mainly chosen in the midrange to the top range of interesting examples. We have thus proven our hypothesis that sense-aware word similarities are indeed beneficial.

3.1.5 Testing Multiple Languages

In (Logacheva et al., 2020), we test the system's performance for multi-language lexical similarity and relatedness tasks. More specifically, we use the SemR-11 datasets[13] (Barzegar et al., 2018), which provide manually annotated word similarity or relatedness scores in 11 different languages. Barzegar et al. (2018) collected the SemR-11 data by canonically translating the English similarity/relatedness benchmarks including Miller & Charles (mc, Miller and Charles, 1991), Rubenstein & Goodenough (rg; Rubenstein and Goodenough, 1965), WordSimilarity-353 (WordSim353; Finkelstein et al., 2001), and Simlex-999 (SimLex999; Hill et al., 2015). We thus evaluate the system for Arabic (ar), German (de), Spanish (es), Farsi (fa), French (fr), Italian (it), Dutch (nl), Portuguese (pt), Russian (ru), Swedish (sv), Chinese (zh), and English (en). We integrated the evaluation into our `senseasim` framework[14] (**Remus** and Biemann, 2018), where the extracted sense inventories and `fastText`[15](Mikolov et al., 2018) embedding vectors for each of the SemR-11 languages and English are used as a basis. One significant advantage of `fastText` is that sub-word units are modeled rather than words. Out-of-vocabulary words (OOVs) are thus unlikely to appear. Grave et al. (2018) provide embedding models for 157 languages, which we use as the basis for testing the 12 languages (Logacheva et al., 2020).

Unfortunately, we do not have the sense inventories as per JoBimText for a large fraction of the 12 languages. In (Logacheva et al., 2020), we thus use a

13. https://github.com/Lambda-3/Gold-Standards/tree/master/SemR-11
14. https://github.com/uhh-lt/senseasim
15. https://fasttext.cc/

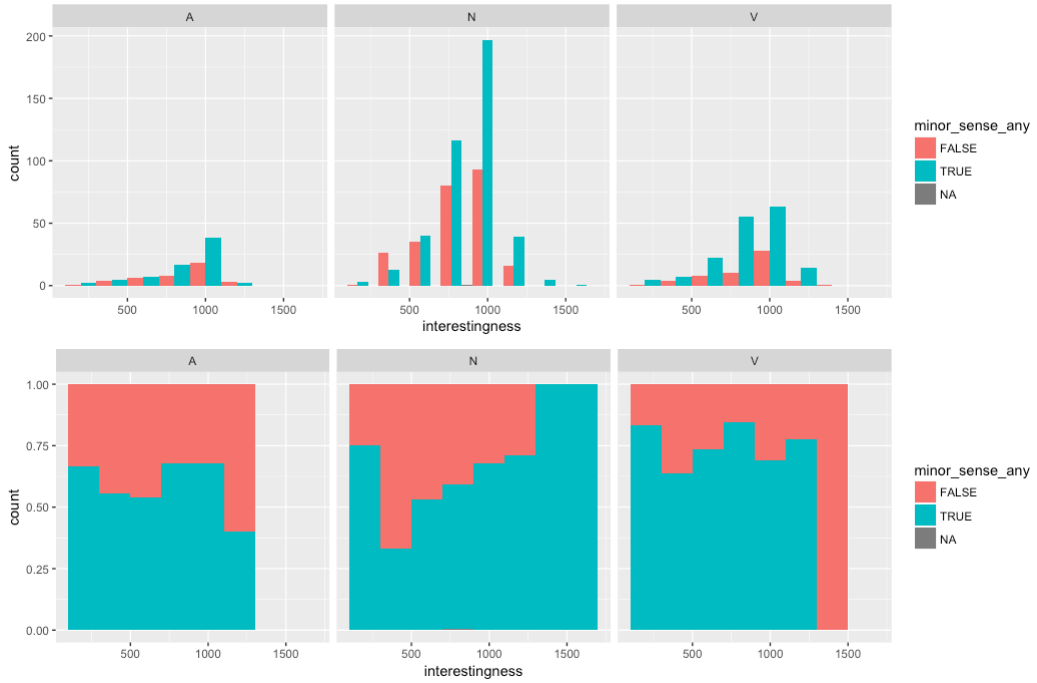


Figure 3.2: Visualization of minor sense selection with respect to our interestingness score. The top row shows for each POS in the SimLex999 dataset a histogram for the number of examples (Y-axis) where at least one word sense was chosen which is not the major sense (biggest cluster) for some interval in the interestingness score (X-Axis). The bottom row shows the relative amounts of the counts in each interval.

slight variation of the methodology presented in Section 3.1.2. As discussed, a major drawback of our clustering methodology used in Section 3.1.2 is the lack of multiple senses in the list of nearest neighbors of a word. A workaround for that might be to increase the number of nearest neighbors to generate or even use the entire vocabulary. This, however, generates too much noise, such that the similarity graph contains too many unrelated words, which has a negative influence on the representation for the clustering algorithm, which will then fail to find good clusterings of neighboring words. In (Logacheva et al., 2020), we thus suggest making use of linguistic regularities in neural embedding models by using vector arithmetics such as addition and subtraction, as described by Mikolov, Yih, et al. (2013). Consider the infamous example $\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} \simeq \overrightarrow{queen}$, where gender is implicitly modeled as a linguistic regularity. The idea is to use vector subtraction to find pairs of dissimilar words and construct the graph only from the words from those so-called 'anti-edges'. The procedure is as follows:

1. For a particular word w, extract a list of N nearest neighbors: $\text{nn}^{+}(w, N) = \{v_1 \dots v_N\}$.

2. Compute a list $\Delta = \{\delta_1 \dots \delta_N\}$, where $\delta_i = w - v_i$. A vector δ_i is supposed to be dissimilar to v_i with respect to w.

3. For a particular word w and its nearest neighbors $i \dots n$, generate $\text{nn}^{--}(w) = \{u_1 \dots u_N\}$, where u_i is the nearest word vector to δ_i. $\text{nn}^{--}(w)$ thus contains words that are highly dissimilar to the aligned nearest neighbors $\text{nn}^{+}(w)$.

For illustration purposes, consider the target word $w = \overrightarrow{python}$, its top similar term $v_1 = \overrightarrow{java}$ and the resulting anti-word $u_i = \overrightarrow{snake}$, which is the nearest word vector to $\delta_1 = \overrightarrow{python} - \overrightarrow{java}$. Together they form an anti-edge $(v_i, u_i) = (\overrightarrow{java}, \overrightarrow{snake})$, i.e., nodes that should not be connected.

4. Construct the graph $G_w = (V, E)$, where $V = \{v : v \in \text{nn}^+(w) \wedge v \in \text{nn}^{--}(w)\}$ is the set of nodes which contain only words which appear in both lists $\text{nn}^+(w)$ and $\text{nn}^{--}(w)$. We thus only add words to the graph which are nearest neighbors and might help to distinguish different senses of w. The set of edges is constructed by $E = \{(v_i, x_j) : x_j \in \text{nn}^+(v_i) \wedge v_i \in V \wedge x_j \in V \wedge x_j \neq u_i\}$, i.e., for each neighbor v_i of w, we generate the K nearest neighbors $x_1 \ldots x_K$ and add an edge (v_i, x_j) if x_j is a node in the graph, i.e., it is also a nearest neighbor of w and in the list of anti-words, but it does not explicitly form an anti-edge with v_i.

Note that the difference between N, the number of nearest neighbors for the target word w, and K, the number of nearest neighbors of v_i, is that the former defines the number of nodes in the graph, and the latter defines the edge density between nodes. Unfortunately, the perfect setting for the parameters N and K depend on the target word w; we thus set $N = K$ and test 50, 100, and 200. After the graph construction, the clustering is performed using the Chinese Whispers algorithm (CW; Biemann, 2006). To reduce computational complexity, we reduce the vocabulary size for each of the 157 languages to the top 100,000 words for pre-computing the sense inventories.

For evaluation, we keep all other parameters aligned with the English benchmark setup. Since we are using `fastText`, we are able to generate a vector for each query word even if it did not appear in the `fastText` vocabulary due to `fastText`'s inherent possibility of using subword information. However, for words not appearing in the pre-computed sense inventory, the `senseasim` system uses the original word vector as a single sense heuristic. Using this back-off strategy, a similarity score can be computed for every word pair in the evaluation datasets, but we note that this also results in the standard cosine similarity if both words of a particular sample pair do not appear in the sense inventory. Details of the coverage of words and word pairs, as well as the average number of senses for each dataset, can be found in Table 3.5. Summarizing, we cover at least 85% of words (minimum) and 97% of word pairs (minimum). This suggests that the sense vectors are the determining factor of the following evaluation results. We follow our previous experiments and use the maximum cosine similarity between any of the sense vectors for each word pair.

Results

Table 3.6 summarizes the results by showing the average difference of correlation scores for each sense inventory to the standard cosine similarity per language: $\phi(language, topK) = \text{mean}(\rho_{topK,language} - \rho_{cos,language})$, where ρ refers to Pearson's correlation score. We summed and averaged the results (last two columns of Table 3.6) and conclude that our methodology provides improved results in the average case. Parameterwise, top200 seems to be the most effective setting with an average improvement of 0.019, which confirms our hypothesis that sense

Table 3.5: Coverage: first sub-column: average number of senses per word; second sub-column: coverage percentage of words; third sub-column: coverage percentage of word-pairs.

		MC			RG			SimLex999			WordSim353			∑		
ar	top100	4.3	95%	100%	4.4	92%	100%	–			4.5	95%	100%	4.4	94%	100%
	top200	4.2	95%	100%	4.1	92%	100%	–			3.5	95%	100%	4.0	94%	100%
	top50	4.2	95%	100%	4.0	92%	100%	–			4.4	95%	100%	4.2	94%	100%
de	top100	3.8	97%	100%	4.0	98%	100%	5.2	96%	99%	5.4	99%	100%	4.6	97%	100%
	top200	2.6	97%	100%	2.7	98%	100%	3.9	96%	99%	4.0	99%	100%	3.3	97%	100%
	top50	4.5	97%	100%	4.9	98%	100%	6.3	95%	99%	6.8	98%	100%	5.6	97%	100%
en	top100	5.3	100%	100%	5.5	100%	100%	6.7	100%	100%	5.6	100%	100%	5.8	100%	100%
	top200	1.1	100%	100%	1.0	100%	100%	12.6	100%	100%	11.3	100%	100%	11.0	100%	100%
	top50	3.4	100%	100%	3.4	100%	100%	3.9	100%	100%	3.2	100%	100%	3.4	100%	100%
es	top100	12.9	100%	100%	11.4	100%	100%	1.2	98%	100%	11.0	96%	100%	11.4	99%	100%
	top200	9.2	100%	100%	8.0	100%	100%	7.4	98%	100%	8.1	96%	100%	8.2	99%	100%
	top50	13.2	100%	100%	12.4	100%	100%	11.8	98%	100%	12.8	96%	100%	12.5	99%	100%
fa	top100	3.8	90%	100%	4.3	93%	100%	–			3.8	93%	99%	4.0	92%	100%
	top200	3.3	90%	100%	3.5	93%	100%	–			3.4	93%	99%	3.4	92%	100%
	top50	3.9	88%	100%	4.5	92%	100%	–			3.9	92%	99%	4.1	91%	100%
fr	top100	12.8	100%	100%	13.0	98%	100%	1.8	95%	99%	12.5	98%	100%	12.3	98%	100%
	top200	9.9	100%	100%	9.9	98%	100%	8.4	95%	99%	9.6	98%	100%	9.5	98%	100%
	top50	13.6	100%	100%	13.4	98%	100%	12.1	95%	99%	13.9	98%	100%	13.2	98%	100%
it	top100	12.9	95%	97%	11.7	95%	98%	1.5	98%	100%	12.7	98%	100%	11.9	97%	99%
	top200	9.8	95%	97%	8.7	95%	98%	7.8	98%	100%	9.4	98%	100%	8.9	97%	99%
	top50	12.6	95%	97%	12.1	95%	98%	11.6	98%	100%	13.6	98%	100%	12.5	97%	99%
nl	top100	7.2	100%	100%	7.2	97%	100%	8.9	97%	100%	9.0	99%	100%	8.1	98%	100%
	top200	5.5	100%	100%	5.2	97%	100%	7.4	97%	100%	7.1	99%	100%	6.3	98%	100%
	top50	7.5	100%	100%	7.1	97%	100%	9.0	97%	100%	8.8	99%	100%	8.1	98%	100%
pt	top100	11.1	100%	100%	1.3	100%	100%	1.1	98%	100%	11.0	99%	100%	1.6	99%	100%
	top200	7.2	100%	100%	6.8	100%	100%	7.4	98%	100%	7.6	99%	100%	7.3	99%	100%
	top50	12.2	100%	100%	11.3	100%	100%	12.0	98%	100%	12.8	99%	100%	12.1	99%	100%
ru	top100	6.2	88%	97%	6.8	86%	98%	5.6	89%	98%	7.7	94%	99%	6.6	90%	98%
	top200	4.8	88%	97%	5.1	86%	98%	4.3	89%	98%	5.7	94%	99%	5.0	90%	98%
	top50	7.5	88%	97%	8.1	86%	98%	6.9	89%	98%	9.3	94%	99%	7.9	90%	98%
sv	top100	5.3	88%	97%	5.7	85%	98%	7.8	95%	99%	8.6	97%	100%	6.9	91%	99%
	top200	5.3	88%	97%	4.7	85%	98%	7.1	95%	99%	7.2	97%	100%	6.1	91%	99%
	top50	5.6	88%	97%	5.3	85%	98%	7.5	95%	99%	7.9	96%	99%	6.6	91%	98%
zh	top100	4.3	100%	100%	4.9	98%	100%	–			4.6	97%	100%	4.6	98%	100%
	top200	2.8	100%	100%	3.2	98%	100%	–			3.3	97%	100%	3.1	98%	100%
	top50	5.2	100%	100%	5.1	98%	100%	–			5.5	96%	100%	5.3	98%	100%

Table 3.6: Averaged difference of Pearson's correlation score for all datasets per language with regard to the cosine similarity baseline.

	ar	de	en	es	fa	fr	it	nl	pt	ru	sv	zh	$\sum$	ϕ
top100	-.004	+.058	-.004	-.004	+.060	+.018	-.003	-.012	+.006	+.060	-.023	+.032	+.183	+.015
top200	+.011	+.065	-.008	+.009	+.080	+.017	-.023	-.010	+.007	+.060	-.014	+.040	+.233	+.019
top50	+.021	+.072	-.005	-.004	+.071	+.017	-.018	-.011	+.014	+.057	-.015	+.026	+.225	+.019

vectors are helpful not only for the English language. However, we also note that while the sense vectors help on average, performance is also hurt for some of the tested languages. This is particularly prevalent for Italian (it), Dutch (nl), and Swedish (sv), where all of the tested inventories perform slightly worse than standard cosine similarity. Also, we found that not a particular inventory setting is clearly beneficial for all languages, although top200 still performs best on average. For completeness, we list the full results in Table 3.7, which shows Pearson's correlation scores of the lexical similarity/relatedness datasets for the 12 languages and parameterized inventories compared to standard cosine similarity.

3.2 Analyzing Sense Modeling Abilities of Contextual Embeddings

Contextualized word embeddings (CWE) such as provided by ELMo (Peters et al., 2018), Flair NLP (Akbik et al., 2018), or BERT (Devlin et al., 2019) were a massive innovation in NLP when they were introduced. In contrast to static word embeddings (SWEs), which provide a single word embedding regardless of their use, CWEs provide a semantic vector representation of a word depending on its current context, thus covering compositionality. Their advantage over static word embeddings has been shown for several tasks, such as text classification, sequence tagging, or machine translation. Since vectors of a particular word vary with the use of the word depending on the current context it is used in, they implicitly provide a model for word sense disambiguation (WSD). In (Wiedemann et al., 2019), we introduced a simple but effective approach to WSD using a nearest neighbor classification using CWEs and analyzed the implicit sense modeling abilities of such CWEs. We compared the performance of different CWE models for the task and report improvements over state-of-the-art approaches for two standard WSD benchmark datasets at the time of writing. We also showed that the pre-trained BERT model can place polysemic words into distinct *sense regions* of the embedding space, while ELMo and Flair NLP do not seem to possess this ability. In the following sections, we present our findings from (Wiedemann et al., 2019).

3.2.1 Synonymy and Polysemy of Word Representations

Lexical semantics is characterized by a high degree of polysemy, i.e., the meaning of a word changes depending on the context it appears in (Harris, 1954). *Word Sense Disambiguation* (WSD) is the task of identifying the sense of a word within its use from a (usually) fixed inventory of senses. For the English language, WordNet (Fellbaum, 1998) is the most commonly used sense inventory, providing more

Table 3.7: Full results of the difference of the Pearson's correlation score for all datasets per language with regard to the cosine similarity baseline. Note that the missing values in SimLex999 are due to missing translations in the SemR-11 dataset.

		MC	RG	SimLex999	WordSim353	$\sum$	ϕ
ar	top100	+.021	-.031	–	-.003	-.012	-.004
	top200	+.042	-.001	–	-.006	+.034	+.011
	top50	+.045	+.010	–	+.007	+.062	+.021
de	top100	+.137	+.078	-.005	+.023	+.233	+.058
	top200	+.147	+.097	-.005	+.020	+.259	+.065
	top50	+.177	+.103	-.012	+.021	+.288	+.072
en	top100	-.028	-.013	+.032	-.007	-.016	-.004
	top200	-.010	-.027	+.015	-.010	-.032	-.008
	top50	-.020	-.022	+.021	-.001	-.022	-.005
es	top100	-.020	-.000	+.012	-.005	-.014	-.004
	top200	+.006	+.016	+.016	-.005	+.034	+.009
	top50	-.012	+.004	+.019	-.026	-.014	-.004
fa	top100	+.079	+.038	–	+.062	+.179	+.060
	top200	+.125	+.048	–	+.065	+.239	+.080
	top50	+.091	+.049	–	+.075	+.214	+.071
fr	top100	+.011	-.021	+.072	+.011	+.073	+.018
	top200	+.004	-.003	+.058	+.010	+.068	+.017
	top50	-.001	-.008	+.077	-.001	+.068	+.017
it	top100	+.002	-.021	-.011	+.017	-.013	-.003
	top200	-.039	-.044	-.012	+.003	-.093	-.023
	top50	-.044	-.008	-.016	-.005	-.074	-.018
nl	top100	-.018	-.032	-.000	+.002	-.049	-.012
	top200	-.013	-.020	-.011	+.004	-.040	-.010
	top50	-.004	-.043	+.001	+.001	-.045	-.011
pt	top100	-.007	+.011	+.015	+.006	+.025	+.006
	top200	+.009	-.004	+.009	+.013	+.027	+.007
	top50	+.018	+.016	+.022	-.000	+.056	+.014
ru	top100	+.180	+.042	-.034	+.052	+.238	+.060
	top200	+.201	+.050	-.053	+.042	+.241	+.060
	top50	+.182	+.043	-.034	+.035	+.226	+.057
sv	top100	-.090	-.014	+.020	-.009	-.094	-.023
	top200	-.075	-.008	+.017	+.010	-.057	-.014
	top50	-.056	-.006	+.012	-.010	-.061	-.015
zh	top100	+.008	+.008	–	+.079	+.095	+.032
	top200	+.017	+.005	–	+.097	+.120	+.040
	top50	+.008	+.004	–	+.065	+.077	+.026

than 200K word-sense pairs.

To train and evaluate WSD systems, a number of datasets have been published, for example, in the SemEval[16] workshop series. In the *lexical sample task*, a training set and a test set is provided, where each sentence contains exactly one annotated word with a sense identifier, and the task is to disambiguate only this one given word (Kilgarriff, 2001; Mihalcea et al., 2004). In the *all-words task*, all (potentially ambiguous) words of a sentence are annotated with a sense identifier, and the task is to predict all words' senses jointly (Edmonds and Cotton, 2001; Snyder and Palmer, 2004). To facilitate the comparison of WSD systems, efforts have been made to provide a comprehensive evaluation framework, such as in (Raganato et al., 2017), and to unify publicly available datasets for the English language, such as in (Vial et al., 2018b).

Word sense disambiguation systems rely on some methodology of context representation to predict the correct sense. The context is typically modeled using dictionary resources linked with senses. Approaches to WSD can be roughly distinguished into three types:

knowledge-based approaches utilize language resources such as dictionaries, thesauri, and knowledge graphs to infer senses and can be implemented with any kind of supervision.

supervised approaches train a (multiclass) classifier to predict a sense given the target word and its context based on the annotated training dataset. However, in (Levy et al., 2015), for example, we argue that supervised distributional approaches do not necessarily learn such kind of relations; they instead learn prototypical patterns in the data, in this case, prototypical senses that simply occur with too many samples in the dataset.

semi-supervised approaches extend manually created training sets by large corpora of unlabeled data to improve WSD performance and coverage.

A fundamental assumption in structuralist linguistics is the distinction between *signifier* and *signified* as introduced by de Saussure in the early 20th century (cf. Section 1.2; de Saussure, 1983; orig. 1916). Computational approaches implicitly assume identity between signifier and signified when using character strings as the only representation of word meaning. Different word senses are collapsed into the same exact string representation. In this respect, word counting and dictionary-based approaches to analyze natural language texts have been criticized as pre-Saussurean (Pêcheux, 2022). In contrast, the *Distributional Hypothesis* (DH; cf. Section 1.2; Harris, 1951) not only states that meaning is dependent on context, but it also states that words occurring in the same contexts tend to have a similar meaning. Hence, a more elegant way of representing meaning is to use the context of a word as an intermediate semantic representation that mediates between signifier and signified. For this, explicit vector representations, such as the bag-of-words, TF-IDF, or latent vector representations – embeddings such as Word2Vec or LSA – have been widely used.

Semantic vector representations allow synonymous terms to occur in the same vicinity in the vector space, which can be exploited for modeling virtually any

16. Formerly known as SensEval.

downstream NLP task. Still, a polysemic term is represented by only a single vector representing all its different senses. To model polysemy as well as synonymy, we introduced the idea of sense embeddings instead of word embeddings (cf. Section 3.1). However, employing sense embeddings in a downstream NLP task requires a reliable WSD system to decide how to choose the appropriate embedding from the sense inventory.

Hence, recent efforts try to model the usage of a sense implicitly by simply employing the context, which makes fixed word sense inventories obsolete for end2end systems. Using contextualized word embeddings (CWE), a word will be composed by a unique vector representation for each unique context it appears in, thus also addressing the idea of vector compositionality within a sentence. The contextualized vector representation is supposed to encode the meaning of a word within the current context. This enables downstream tasks to actually distinguish the two levels of signifier and signified, allowing more realistic modeling of natural language. The advantage of such contextually embedded token representations compared to static word embeddings has been shown for a number of tasks, such as text classification (Zampieri et al., 2019) or sequence tagging (Akbik et al., 2018), among many more. In the following, we distinguish between the terms *type* and *token*. The term type is used to refer to a vocabulary entry of a word, and the term token is used to refer to a word in its use, i.e., within a sentence or phrase.

In (Wiedemann et al., 2019), we show that CWEs can be utilized directly to approach the WSD task due to their nature of providing distinct vector representations for the same type, i.e., for a token. To analyze the semantic capabilities of CWEs, we employ a simple yet interpretable approach to WSD using a k-nearest neighbor classification (kNN) approach. We compare the performance of three different CWE models on four standard benchmark datasets. Our evaluation yields that not all contextualization approaches are equally effective in dealing with polysemy and that the simple kNN approach suffers severely from sparsity in training datasets. Yet, by using kNN, we can include provenance into our model, which allows for investigating the training sentences that lead to the classifier's decision. Thus, we are able to study to what extent polysemy is captured by a specific contextualization approach. At the time of writing, we were even able to report new state-of-the-art (SOTA) results for two datasets. The following section introduces approaches to WSD based on neural network architectures and approaches to contextualized word embeddings.

3.2.2 Neural Word Sense Disambiguation

Several efforts have been made to induce vectors for the multiplicity of senses a word can express. Bartunov et al. (2016), Neelakantan et al. (2014), or Rothe and Schütze (2015) induce so-called sense embeddings in a pre-training fashion. While Bartunov et al. (2016) induce sense embeddings in an unsupervised way and only fix the maximum number of senses per word, Rothe and Schütze (2015) requires a pre-labeled sense inventory such as WordNet. Other approaches include the reuse of pre-trained word embeddings to induce new sense embeddings, such as SenseGram (Pelevina et al., 2016) or senseasim (cf. Sec. 3.1, **Remus** and Biemann, 2018). In (Logacheva et al., 2020), we also use induced sense embeddings for the downstream task of WSD similar to Panchenko, Marten, et al. (2017). An

overview of further word sense modeling approaches can be found in (Camacho-Collados and Pilehvar, 2018).

With regard to our approach, Melamud et al. (2016) and Yuan et al. (2016) are most related, i.e., they both compute sentence context vectors for ambiguous target words and select nearest neighbors of context vectors to determine the target word sense in the prediction phase. Yuan et al. (2016) additionally use unlabeled sentences with label propagation in a semi-supervised setting to overcome the sparsity issue within the training data. Further, Kågebäck and Salomonsson (2016) employ a recurrent neural network (RNN) to classify sense labels for ambiguous target words given their contexts. Contrary to many previous approaches, which rely on feature engineering, Taghipour and Ng (2015) use only pre-trained GloVe word embeddings by (Pennington et al., 2014) to achieve SOTA results on two English lexical sample datasets. For the all-words WSD task, Vial et al. (2018a) employ a recurrent neural network where they sequentially classify sense labels for all tokens within a sentence. They also introduce an approach to collapse the sense vocabulary from WordNet to unambiguous hypersenses, which increases the label-to-sample ratio for each label (sense identifier). By training their system on the large sense annotated datasets SemCor[17] (Miller et al., 1993) and the Princeton WordNet Annotated Gloss Tags Corpus[18] based on WordNet synset definitions (Fellbaum, 1998), they achieve SOTA results on most all-words WSD benchmarks. A similar architecture with an enhanced sense vocabulary compression was applied in (Vial et al., 2019), but instead of static GloVe embeddings, contextualized BERT wordpiece embeddings (Devlin et al., 2019) are used as input for training, where particularly the BERT embeddings improved the overall performance.

3.2.3 Contextualized Word Embeddings

For most downstream NLP tasks, using CWEs drastically improved the performance of neural architectures compared to SWEs. However, the contextualization methodologies are different. Because of that, we hypothesize that they are also different in their ability to capture polysemy.

Like static word embeddings, CWEs are pre-trained on large amounts of unlabeled data by a language modeling objective. Because of their popularity, we investigate three widely applied approaches: *Flair* (Akbik et al., 2018), *ELMo* (Embeddings from Language Models; Peters et al., 2018), and *BERT* (Bidirectional Encoder Representations from Transformers; Devlin et al., 2019).

Flair: For contextualizing a particular word vector, Akbik et al. (2018) concatenate the static pre-trained word embedding vector of a word, e.g., the GloVe word embeddings (Pennington et al., 2014), with the left and right neighboring word vectors, the context. The context vectors are computed by two RNNs, one character language model trained from left to right and one from right to left. Their approach has been applied successfully for sequence tagging tasks such as named entity recognition and part-of-speech tagging.

17. http://web.eecs.umich.edu/~mihalcea/downloads.html#semcor
18. https://wordnetcode.princeton.edu/glosstag.shtml

ELMo: Peters et al. (2018) approaches contextualization similarly to Flair, but instead of two character language models, two stacked RNNs for words are trained, again one from left to right and another from right to left. For CWEs, outputs from the embedding layer and the two bidirectional recurrent layers are collapsed into one layer by a weighted, element-wise addition.

BERT: In contrast to the previous two approaches, Devlin et al. (2019) provide contextualized wordpiece embeddings in an end-to-end language model architecture. For this, a self-attention based Transformer architecture is used, which, in combination with a masked language modeling objective, allows the model to 'see' all left and right context wordpieces of a target word at the same time. Self-attention (Vaswani et al., 2017) and thus non-directionality of the language modeling objective result in substantial performance gains compared to previous approaches.

Next, we present our experiments and results from (Wiedemann et al., 2019) and investigate our hypothesis that contextual embeddings can identify polysemy using the contextualized models.

3.2.4 Nearest Neighbor Classification for WSD

We approach WSD by using k-nearest neighbor classification (kNN) to investigate the semantic properties of contextualized word embeddings. Compared to other classification approaches such as *support vector machine*s (SVMs) or neural networks (NNs), kNN has the advantage that it can be used to investigate the training examples that lead to a certain decision.

In its simplest form, the kNN classification algorithm assigns a label to an unseen test sample based on the majority of the assigned labels of the k-nearest training samples in its vicinity based on some distance metric (Cover and Hart, 1967). Although complex weighting schemes for kNN exist, we stick to the non-parametric version of the algorithm to be able to investigate the semantic properties of the individual contextual embedding approaches. As a distance measure for kNN, we rely on the cosine distance of the contextual embeddings. The cosine distance is the inverse of the cosine similarity. Note that any similarity metric can be converted to a dissimilarity (distance) metric by means of inversion. The approach considers only senses for a target word observed during training. We use spaCy[19] (Honnibal and Johnson, 2015) for pre-processing and the lemmatized form of a word.

Since BERT uses wordpieces, i.e., subword units of words, we re-tokenize the sentence with BERT's wordpiece tokenizer and average all sub-word CWEs that belong to the target word or phrase. Moreover, for the experiments with BERT embeddings[20], we follow the heuristic by Devlin et al. (2019) and concatenate the averaged wordpiece vectors of the last four layers.

We test different values for our single hyperparameter $k \in \{1 \dots 10, 50, 100, 500, 1000\}$. Like words, word senses also follow a power-law distribution, which makes

19. https://spacy.io/

20. We use the `bert-large-uncased` model.

Table 3.8: Properties of the used datasets. For the test sets (Te), we do not report k' since they are not used as kNN training instances.

	SE-2 (Tr)	SE-2 (Te)	SE-3 (Tr)	SE-3 (Te)	S7-T7 (coarse)	S7-T17 (fine)	SemCor	WNGT
#sentences	8,611	4,328	7,860	3,944	126	245	37,176	117,659
#CWEs	8,742	4,385	9,280	4520	455	6,118	230,558	1,126,459
#distinct words	313	233	57	57	327	1,177	20,589	147,306
#senses	783	620	285	260	371	3,054	33,732	206,941
avg #senses p. word	2.50	2.66	5.00	4.56	1.13	2.59	1.64	1.40
avg #CWEs p. word & sense	11.16	7.07	32.56	17.38	1.23	2.00	6.83	5.44
avg k'	2.75	-	7.63	-	-	-	3.16	2.98

simple baseline approaches for WSD, like the *most frequent sense* (MFS) baseline, strong competitors, and hard to beat. Other effects of the skewed distribution are imbalanced training sets. Many senses described in WordNet only have one or two example sentences in the training sets or are not present at all. This is severely problematic for larger k because the majority class dominates the neighborhood of a word. To counter this effect a little, we modify the majority voting of kNN to $k' = min(k, |V_s|)$ where V_s is the set of CWEs with the least frequent training examples for a given word sense s. Larger values of k are thus only used if the number of training examples support this and are reduced dynamically if they do not show this support.

3.2.5 Datasets

We conduct our experiments with the help of four standard WSD benchmark datasets, two lexical sample tasks and two all-words tasks. The lexical sample tasks, SensEval-2 (SE-2; Kilgarriff, 2001) and SensEval-3 (SE-3; Mihalcea et al., 2004) provide a training and test set each. The all-words tasks of SemEval 2007 Task 7 (S7-T7; Navigli et al., 2007) and Task 17 (S7-T17; Pradhan et al., 2007) provide only test data and both contain a substantial overlap of documents with each other. The two sets differ in granularity, though. While ambiguous terms in Task 17 are annotated only with their exact WordNet synset id, in Task 7, ambiguous terms are annotated additionally with WordNet synset ids coming from traversing hypernymy relations. Labels in Task 7 are thus considered coarser and less fine-grained than in Task 17.

For training a system for the all-words tasks S7-T7 and S7-T17, we employed *a*) the SemCor dataset (Miller et al., 1993), and *b*) the Princeton WordNet Annotated Gloss Tags corpus (WNGT; Fellbaum, 1998) separately to investigate the influence of different training sets on our approach. For all experiments, we utilized the mentioned datasets provided by the UFSAC framework[21] (Vial et al., 2018b). An overview of the data, including some statistics, can be found in Table 3.8. The table shows that the SE-2 and SE-3 training datasets provide many more examples per word and sense than SemCor or WNGT.

21. Unification of Sense Annotated Corpora and Tools. We used version 2.1: https://github.com/getalp/UFSAC

Table 3.9: kNN with $k = 1$ WSD performance (F1%). The best results for each test set are marked in bold.

Model	SE-2	SE-3	S7-T7 (coarse)		S7-T17 (fine)	
			SemCor	WNGT	SemCor	WNGT
Flair	65.27	68.75	69.24	78.68	45.92	50.99
ELMo	67.57	70.70	70.80	79.12	52.61	50.11
BERT	**76.10**	**78.62**	73.61	**81.11**	**59.82**	55.16

3.2.6 Evaluation

We conduct two experiments to determine whether contextualized word embeddings can be used to address WSD. In our first experiment, we compare different pre-trained embeddings with $k = 1$. In our second experiment, we test multiple values of k and use only BERT embeddings and WNGT for S7-T7 and Semcor for S7-T17, since – spoiler ahead – this combination performed best in the first experiment. By testing multiple values of k, we attempt to estimate and report an optimal value for k. Further, we examine the results qualitatively to analyze typical successful predictions, i.e., true positives (TP), and typical error cases, i.e., false positives (FP), of our approach.

Testing CWEs

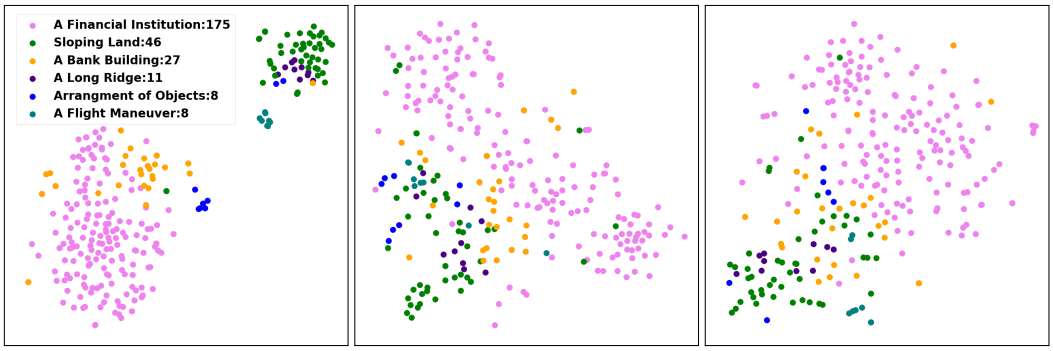
To compare different CWE approaches, we use $k = 1$ nearest neighbor classification. Table 3.9 shows the results have a high variance in performance. Simple kNN with ELMo or BERT embeddings yield the best results, which, at the time of writing, happened to be state-of-the-art on the lexical sample task SE-2 (cf. also Table 3.10). Also, BERT embeddings substantially outperform other model embeddings. For S7-T7 and S7-T17, the content and structure of the out-of-domain (OOD) SemCor and WNGT training datasets differ drastically from those in the test data. In fact, the similarity of the results regarding contextualized embeddings largely relies on semantically and syntactically similar contexts of polysemic target words. Hence, the more example sentences can be used for a sense, the higher are the chances that a nearest neighbor expresses the same sense. As can be observed from Table 3.8, the SE-2 and SE-3 training datasets provide more CWEs for each word and sense, and our approach performs better with an increasing number of CWEs, even with a higher average number of senses per word than in SE-3. We thus conclude that the nearest neighbor approach suffers specifically from data sparseness. Other aspects of the target word, such as syntax, also strongly influence the kNN decision. This is probably due to the fact that the different layers of, e.g., BERT, can be accounted for different properties of language, such as syntax, grammar, semantics on various levels, etc.

Optimizing k

To obtain more robust nearest neighbor classification and potentially even better results, we optimized the hyperparameter k. Table 3.10 shows our best results using BERTs embeddings combined with results from related work. For these experiments, we focused on the best-performing setups from Table 3.9, hence

Table 3.10: Best kNN models vs. most frequent sense (MFS) and state of the art (F_1). The best results are in bold, the previous SOTA is in italics, and our best results are underlined.

k	SE-2	SE-3	S7-T7 (WNGT)	S7-T17 (SemCor)
1	76.10	78.62	81.11	59.82
2	76.10	78.62	81.11	59.82
3	**76.52**	79.66	80.94	59.38
4	**76.52**	79.55	80.94	59.82
5	76.43	79.79	81.07	60.27
6	76.43	79.81	81.07	60.27
7	76.50	80.02	81.03	60.49
8	76.50	79.86	81.03	60.49
9	76.40	79.97	81.03	60.49
10	76.40	**80.12**	81.03	60.49
50	76.43	79.66	81.11	60.94
100	76.43	79.63	81.20	60.71
500	76.38	79.63	81.11	60.71
1000	76.38	79.63	81.11	60.71
MFS	54.79	58.95	70.94	48.44
Kågebäck et al. (2016)	*66.90*	*73.40*	-	-
Yuan et al. (2016)	-	-	84.30	63.70
Vial et al. (2018a)	-	-	86.02	66.81
Vial et al. (2019)	-	-	***90.60***	***71.40***



(a) BERT (b) Flair (c) ELMo

Figure 3.3: t-SNE plots of different senses of 'bank' and their contextualized embeddings. The legend briefly describes the respective WordNet synsets and the frequency of occurrence in the training data. Here, the SE-3 training dataset is used.

BERT embeddings and WNGT for SE-T7 and SemCor for SE-T17. For SE-2 and SE-3, we achieve new state-of-the-art results using $k = 3$ for SE-2 and $k = 10$ for SE-3.[22] We observe convergence with higher k values since our k' normalization heuristic dominates. We also achieve minor improvements for both S7-T* datasets with a higher k ($k = 100$ and $k = 50$, respectively).

22. At the time of writing.

Visualizing Senses in the CWE Vector Space

A natural question arises: How well do different CWE models encode sense information in their vector space? We use t-SNE (T-distributed stochastic neighbor embedding; Maaten and Hinton, 2008) to visualize contextual word embeddings in two dimensions. Figure 3.3 shows t-SNE plots of six different senses of the word 'bank' in the SE-3 training dataset encoded by the three different CWE methods. For visualization purposes, we exclude senses with a frequency of less than two. Although the t-SNE process loses information, we can clearly see that Flair embeddings hardly allow any separation of clusters as most senses are scattered across the entire plot. In the ELMo embedding space, the major senses are slightly more separated in different regions of a single cloud. Visually, senses are clearly separated as distant clusters in the BERT embedding space, where related senses are still nearby, e.g., 'bank' as a financial institution and 'bank' as a building are still somewhat related; unrelated senses, however, are far away, e.g., the financial institution versus the river bank. Also, within the larger clusters, single senses are spread mostly in separated regions of the cluster. Thus, we conclude that BERT embeddings do encode some form of sense knowledge, which also explains why kNN performs so much better with BERT than with the other approaches. Still, a more powerful non-linear classification approach such as the one presented by Vial et al. (2019) can learn clearer decision boundaries in higher dimensions. Such clear decision boundaries seem to solve the data sparseness issue of kNN successfully.

Error analysis

By analyzing the true positive (TP) and false positive (FP) predictions from a qualitative point of view, we can infer some semantic properties of BERT embeddings within the used training corpora. Table 3.11 shows selected examples of polysemic words in different test sets, including their nearest neighbor from the respective training set.

For instance, the word overlap within the contexts leads to correct predictions, as can be seen in 'along the *bank* of the river' and 'along the *bank* of the river Greta' (2). Another example is 'little earthy bank' and 'huge bank [of snow]' (3), where even semantically similar context words ('little' and 'huge') overlap. On the other hand, vocabulary overlap and semantic relatedness, as in 'land bank' (5), can also lead to false predictions. Another interesting example of the latter is the confusion between 'grass bank' and 'river bank' (6), where the nearest neighbor sentence in the training set shares some military context ('grenade' and 'gun') with the target sentence. The correct sense **bank**$_{\text{[Sloping Land]}}$ and the predicted sense **bank**$_{\text{[A Long Ridge or Pile (of earth)]}}$ share a strong semantic similarity, too. In the example, they could even be used interchangeably.

In Example (10), the target sense is an action, i.e., a verb sense, while the predicted sense is a noun. In general, this could be easily fixed by restricting the classifier decision to the desired syntactic class. On the other hand, consider Example (12): Although it is a false positive, it shows nicely that the approach is able to find semantically and syntactically similar nearest neighbors even though BERT never learned syntactic classes explicitly. This effect has been investigated in-depth by Jawahar et al. (2019), who found that each BERT layer

Table 3.11: Example predictions based on nearest neighbor sentences. The word in question is marked in boldface, subset with a short description of its WordNet synset (true positives green, false positives red).

	Example Sentence SE-3 (test)	Nearest Neighbor SE-3 (train)
(1)	President Aquino, admitting that the death of Ferdinand Marcos had sparked a wave of sympathy for the late dictator, urged Filipinos to stop weeping for the man who had laughed all the way to the **bank**[A Bank Building].	They must have been filled in at the **bank**[A Bank Building] either by Mr Hatton himself or else by the cashier who was attending to him.
(2)	Soon after setting off we came to a forested valley along the **banks**[Sloping Land] of the Gwaun.	In my own garden the twisted hazel, corylus avellana contorta, is underplanted with primroses, bluebells and wood anemones, for that is how I remember them growing, as they still do, along the **banks**[Sloping Land] of the rive Greta
(3)	In one direction only a little earthy **bank**[A Long Ridge] separates me from the edge of the ocean, while in the other the valley goes back for miles and miles.	The lake has been swept clean of snow by the wind, the sweepings making a huge **bank**[A Long Ridge] on our side that we have to negotiate.
(4)	However, it can be possible for the documents to be signed after you have sent a payment by cheque provided that you arrange for us to hold the cheque and not pay it into the **bank**[A Financial Institution] until we have received the signed deed of covenant.	The purpose of these stubs in a paying – in book is for the holder to have a record of the amount of money he had deposited in his **bank**[A Bank Building].
(5)	He continued: assuming current market conditions do not deteriorate further, the group, with conservative borrowings, a prime land **bank**[A Financial Institution] and a good forward sales position can look forward to another year of growth.	Crest Nicholson be the exception, not have much of a land **bank**[Supply or Stock] and rely on its skill in land buying.
(6)	The marine said, get down behind that grass **bank**[A Long Ridge], sir, and he immediately lobbed a mills grenade into the river.	The guns were all along the river **bank**[Sloping Land] as far as I could see.
	S7-T17	SemCor
(7)	Some 30 **balloon**[Large Tough Nonrigid Bag] shows are held annually in the U.S., including the world's largest convocation of ersatz Phineas Foggs – the nine-day Albuquerque International Balloon Fiesta that attracts some 800,000 enthusiasts and more than 500 balloons, some of which are fetchingly shaped to resemble Carmen Miranda, Garfield or a 12-story-high condom.	Homes and factories and schools and a big wide federal highway, instead of peaceful corn to rest your eyes on while you tried to rest your heart, while you tried not to look at the **balloon**[Large Tough Nonrigid Bag] and the bandstand and the uniforms and the flash of the instruments.
(8)	The condom **balloon**[Large Tough Nonrigid Bag] was denied official entry status this year.	Just like the **balloon**[Large Tough Nonrigid Bag] would go up and you could sit all day and wish it would spring a leak or blow to hell up and burn and nothing like that would happen.
(9)	Starting with congressman Mario Biaggi (now serving a jail **sentence**[The Period of Time a Prisoner Is Imprisoned]), the company began a career of bribing federal, state and local public officials and those close to public officials, right up to and including E. Robert Wallach, close friend and adviser to former attorney general Ed Meese.	When authorities convicted him of practicing medicine without a license (he got off with a suspended **sentence**[The Period of Time a Prisoner Is Imprisoned] of three years because of his advanced age of 77), one of his victims was not around to testify: he was dead of cancer."
(10)	Americans it seems have followed Malcolm Forbes's hot-air lead and taken to **balloon**[To Ride in a Hot-Air Balloon] in a heady way.	Just like the **balloon**[Large Tough Nonrigid Bag] would go up and you could sit all day and wish it would spring a leak or blow to hell up and burn and nothing like that would happen.
(11)	Any question as to why an author would believe this plaintive, high-minded note of assurance is necessary is answered by reading this **book**[A Published Written Work] about sticky fingers and sweaty scammers.	But the **book**[A Written Version of a Play] is written around a somewhat dizzy cartoonist, and it has to be that way.
(12)	In between came lots of coffee drinking while **watching**[To Look Attentively] the balloons inflate and lots of standing around deciding who would fly in what balloon and in what order [...].	So Captain Jenks returned to his harbor post to **watch**[To Follow With the Eyes or the Mind; observe] the scouting plane put in five more appearances, and to feel the certainty of this dread rising within him.

learns different structural aspects of language. Example (12) also emphasizes the difficulty of distinguishing verb senses because verb senses are very fine-grained in WordNet and thus harder to distinguish even by humans, i.e., in the example, the correct label is **watch**[look attentively] whereas the nearest neighbor is

Table 3.12: Best POS-sensitive kNN models. Bold numbers are improved results over Table 3.10.

k	SE-2	SE-3	S7-T7		S7-T17	
			SemCor	WNGT	SemCor	WNGT
1	76.10	78.62	79.30	85.23	61.38	61.98
3	76.52	79.66	**79.44**	85.01	60.94	**62.64**
7	76.50	80.02	79.35	85.05	62.50	62.20
10	76.40	80.12	79.40	85.10	62.72	62.20
30	76.43	79.66	79.40	85.14	**63.17**	61.98
70	76.43	79.61	79.35	85.23	62.95	61.98
100	76.43	79.63	79.35	**85.32**	62.95	61.98
300	76.43	79.63	79.35	**85.32**	62.95	61.98

Table 3.13: Percentage of senses with a certain POS tag in the corpora.

	Noun	Adj	Verb	Other	avg #POS per word
SE-2 (tr)	41.32	16.57	42.11	0.00	1.0
SE-2 (te)	40.98	16.56	42.46	0.00	1.0
SE-3 (tr)	46.45	3.94	49.61	0.00	1.0
SE-3 (te)	46.17	3.98	49.86	0.00	1.0
S7-T7	49.00	15.75	26.14	9.11	1.03
S7-T17	34.95	0.00	65.05	0.00	1.01
SemCor	38.16	14.70	38.80	8.34	1.10
WNGT	57.93	21.57	15.55	4.96	1.07

watch$_{\text{[follow with the eyes or the mind; observe]}}$. Those senses are strongly related, and the latter sounds like a definition of the former.

POS-informed Prediction

Addressing the issue of mixed POS senses, we ran a further experiment in which we restricted words to their lemma **and** their POS tag. Table 3.12 confirms our hypothesis that including the POS restriction increases the F1 scores for S7-T7 and S7-T17. The results are roughly aligned with the number of different POS tags in the corpora (cf. Table 3.13). The results for SE-2 and SE-3 did not change drastically, which can be explained by the average number of POS tags a particular word is labeled with, i.e., in SE-2 and SE-3, no POS overlap exist between different word classes, senses are only labeled within the same POS class. For SE-7*, SemCor, and WNGT, this property has a stronger variance.

This chapter is based on the following published work by the author and collaborators:

Omer Levy, **Steffen Remus**, Chris Biemann, and Ido Dagan. 2015. Do Supervised Distributional Methods Really Learn Lexical Inference Relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 970–976. Denver, CO, USA.

Varvara Logacheva, Denis Teslenko, Artem Shelmanov, **Steffen Remus**, Dmitry Ustalov, Andrey Kutuzov, Ekaterina Artemova, Chris Biemann, and Alexander Panchenko. 2020. Word sense disambiguation for 158 languages using word embeddings only. In *Proceedings of The 12th Language Resources and Evaluation Conference,* 5943–5952. Marseille, France.

Steffen Remus and Chris Biemann. 2018. Retrofitting Word Representations for Unsupervised Sense Aware Word Similarities. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018),* 1035–1041. Miyazaki, Japan.

Gregor Wiedemann, **Steffen Remus**, Avi Chawla, and Chris Biemann. 2019. Does BERT Make Any Sense? Interpretable Word Sense Disambiguation with Contextualized Embeddings. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019): Long Papers,* 161–170. Erlangen, Germany.

4
Retrieval of Semantic Relations

Pre-trained language models based on the Transformer neural network architecture quickly became state of the art in machine learning to numerous natural language processing (NLP) tasks. In these models, natural language text is represented as a sequence of subtokens called wordpieces. Due to the attention layers in the transformer architecture, wordpieces are embedded into a context-sensitive semantic vector space (Vaswani et al., 2017). In contrast to static word embeddings (SWE), such as Word2Vec, where a token is always aligned with the same vector representation, these contextualized embeddings (CWEs) allow more complex, compositionally aligned, and thus disambiguated semantic representations. In this chapter, which is aligned with the findings in (**Remus** et al., 2022), we investigate the semantic capabilities of pre-trained contextualized word embeddings (CWEs), such as BERT (Devlin et al., 2019), in combination with lexical, syntactical or grammatical information, to perform semantic retrieval of semantic relations for various datasets, i.e., given a query, the goal is to find more instances with the same encoded semantic information. We show that the aggregation of task-agnostic CWE representations with different strategies along syntactic or grammatical dimensions allows for highly precise retrieval and competitive simple nearest neighbor classification of complex categories such as semantic relations. In (**Remus** et al., 2022), we even extend the list of use cases to a number of tasks of varying complexity, such as coarse and fine-grained named entity recognition, short text retrieval, and classification, and retrieval and classification of verb frames, next to the semantic relation retrieval and classification as presented in this work.

4.1 Introduction

Neural language models (NLMs) producing contextualized word embeddings (CWEs) such as BERT (Bidirectional Encoder Representations from Transformers; Devlin et al., 2019), or its many successors, ELMo (Embeddings from Language Models; Peters et al., 2018), or FLAIR (Akbik et al., 2018) are a huge improvement factor for multiple NLP tasks. One major reason for this is the fact that NLMs

can produce a composed vector space representation of a word (or subtoken) based on the sequential context it is currently appearing in, thus representing its compositional meaning. CWEs implicitly disambiguate the meaning of a word (cf. Section 3.2) up to a certain degree, such that, for example, sequence tagging models are able to distinguish two identical surface forms (types) when used in different contexts. For example, both instances of the words 'can' and 'open' in the following two sentences '`Alice opens the can.`' and '`Alice can open the box.`' will be represented with distinct locations in the vector space. Whereas the two associated vectors for the word 'open' are expected to be very similar, the two vectors for the word 'can' are expected to be inherently different, indicating a strong syntactic and semantic shift. In a way, the word 'can' from the first sentence is supposed to be located in the same vicinity as the word 'box' from the second sentence due to the properties of the distributional hypothesis (DH; Harris, 1954).

However, Htut et al. (2019) show that, although certain dependency relations are implicitly encoded in BERT, no equivalent to holistic parsing of syntactical or grammatical structures can be inferred from BERT's attention mechanism. We thus hypothesize that downstream NLP tasks benefit from exploiting explicit syntactical and grammatical cues derived from linguistic knowledge in addition to the contextualized embeddings. To investigate this hypothesis, we define a set of linguistic patterns for aggregating CWEs along linguistically informed dimensions. However, instead of well-researched classification problems in the supervised learning paradigm, we aim to efficiently find semantically similar sequences in a given dataset. To investigate the capabilities of CWEs in such an information retrieval scenario, we refrain from evaluating complex classifier architectures and fine-tuning models. Instead, we focus on retrieval evaluation measures and simple nearest neighbor classification to properly assess the quality of CWEs regarding linguistic patterns. Our retrieval evaluation shows that CWE-based retrieval finds semantically similar sequences with a higher precision than SWE-based retrieval, as produced by, e.g., Word2Vec (Mikolov, Chen, et al., 2013) or GloVe (Pennington et al., 2014). More importantly, we also show in our experiments that, depending on the task, the CWE-based retrieval profits from incorporating additional explicit linguistic knowledge.

In summary, this chapter's contribution is threefold: *a*) we define and use structural linguistic information for different aggregation strategies; *b*) we evaluate those strategies in a retrieval setup to find semantically similar sequences as defined by the particular downstream task; *c*) we evaluate a retrieval-inspired k-nearest neighbor classification approach to validate downstream tasks in a classification framework.

4.2 Related Work

Strubell et al. (2018) showed the benefit of injecting syntactic information into a neural network using self-attention for multi-task learning and called their approach LISA (linguistically-informed self-attention). LISA was applied for dependency parsing, part-of-speech tagging, predicate detection, and semantic role labeling, where the results for all tasks showed significant improvements over previous SOTA, particularly when using contextualized embeddings provided

by ELMo (Peters et al., 2018). In (Wiedemann et al., 2019) and Section 3.2, we showed that contextualized embeddings, particularly BERT (Devlin et al., 2019) inherit a strong degree of sense representation, i.e., polysemous words appear in different areas of the embedding space depending on their use. Wang et al. (2019) implement Elman (1990)'s theory, which says that neural language models are sensitive to word order regularities in simple sentences by explicitly exploiting the inner-sentence structure (word-level ordering) and inter-sentence structure (sentence-level ordering) as training objectives. Wang et al. (2019) argue that their StructBERT model successfully captures the structure of sentences during pre-training. This is achieved by pre-shuffling trigrams that are not used as masked tokens within sentences and using three different sentence prediction objectives: next sentence prediction, previous sentence prediction, and random sentence prediction.

Htut et al. (2019) and Clark et al. (2019) analyze to which extent the attention layers within BERT's encoder stack can be exploited to infer linguistic knowledge. Both works conclude that some attention layers specialize in syntactic structure. Wu et al. (2020) use this fact and specifically measure the impact one word has on another in a sentence by using their so-called perturbed masking technique. Wu et al. (2020) are able to derive a syntax tree from this word-to-word representation matrix. Baldini Soares et al. (2019) used a so-called masking technique to force the model to learn named entity locations within a sequence. By doing so, specific representations for particular relations within text can be learned.

Specifically related to the information retrieval part of this line of work is SBERT (SentenceBERT; Reimers and Gurevych, 2019). SBERT is an extension to pre-trained transformer architectures such as BERT or RoBERTa, which is specifically targeted for sentence similarity search, i.e., finding similar sentences by using cosine similarity, while drastically decreasing search time of standard cosine similarity. SBERT outperforms most other embedding strategies for multiple sentence similarity tasks. However, SBERT requires labeled data in the form of similar and dissimilar sentences, and the training data also defines the architecture of the siamese network, which disqualifies the evaluation of SBERT in our scenario.

4.3 Retrieval by Linguistic Patterns

We approach the retrieval of semantically similar sequences by using linguistic patterns as a nearest neighbor problem, which can be formally described as follows: Let $S := [s_1, \dots, s_n]$ be a training dataset with n instances, where s_i represents a sequence that contains a particular relation, and let $\mathbf{y} = [y_1, \dots, y_n]$ be their corresponding class labels. Instances are decomposed into a set of finer-grained structures by applying linguistic patterns, such as tokens, chunks or dependency paths, etc., which we then use for aggregation of the respective CWEs and as keys for an inverted index, with the sequence and its label as value. Retrieval of a query sequence can then be performed by the same decomposition and CWE aggregation strategy and then looking for the nearest neighbors in the inverted index. More formally, each instance s_i is decomposed into a unique set of m^i structures by using a particular linguistic pattern:

$$s_i \mapsto \{ \mathbf{x}_i^1, \dots, \mathbf{x}_i^{m^i} \}, \tag{4.1}$$

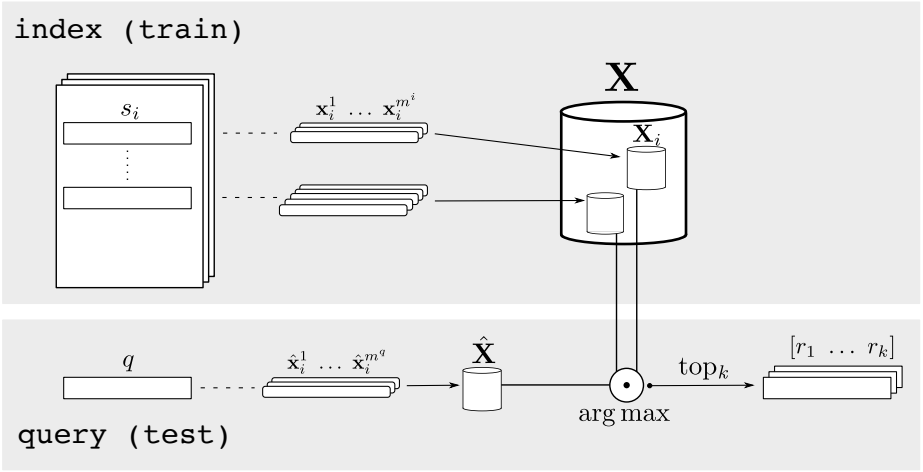


Figure 4.1: Overview of the indexing (training) and retrieval (testing) process.

and its y_i label is replicated accordingly:

$$s_i \mapsto \ \mathbf{X}_i \ := \{ \, y_i^1, \dots, y_i^{m^i} \, \} \, . \tag{4.2}$$

Further, let $\mathbf{x}_i^j$ represent a single aggregated embedding extracted by applying a particular pattern and aggregated by the resulting structure's constituent CWEs. For example, it could be the representation of the actual sequence using an average of all wordpiece embeddings (bag-of-word approach), or it could be the word embedding of a single token in s_i. In the former case, m^i equals 1; in the latter case, m^i equals the number of tokens in s_i. We call $\mathbf{x}_i^j$ a *structured embedding*, refer to $\mathbf{X}_i$ as the set of all structured embeddings extracted from a particular sentence s_i, and define $\mathbf{X}$ to be the entire index of training examples, i.e., the superset of all structured embedding sets $\mathbf{X}_i$ collected across all training sentences $s_i \dots s_n$.

During test time, the goal is to retrieve the k most relevant unique sequences $[r_1, \dots, r_k]$ and their target labels for a given query sequence q from the index $\mathbf{X}$ by applying the same linguistic pattern and aggregation strategy that was used for indexing:

$$q \mapsto \hat{\mathbf{X}} \ := \{ \, \hat{\mathbf{x}}^1, \dots, \hat{\mathbf{x}}^{m^q} \, \} \tag{4.3}$$

$$[\, r_1, \dots, r_k \,] \ := \ \underset{\mathbf{X}_i \in \mathbf{X}}{\text{top}_k} \{ \ \underset{\substack{h \in \{1,\dots,m^q\} \\ j \in \{1,\dots,m^i\}}}{\arg\max} \{ \ \text{sim}(\hat{\mathbf{x}}^h, \mathbf{x}_i^j) \ \} \ \} \, , \tag{4.4}$$

where top_k is defined as a function that selects the top k indices as an ordered list from the entire set of training instances regarding their maximum similarity score of one of their respective structured embeddings. The sim function is defined to be a similarity function for two feature vectors, where we chose to use cosine similarity for our experiments. For evaluation purposes, we consider r_i a relevant retrieved instance iff y_{r_i} equals the class label of the test instance q. The entire process of indexing and querying (training and testing, respectively) is illustrated in Figure 4.1.

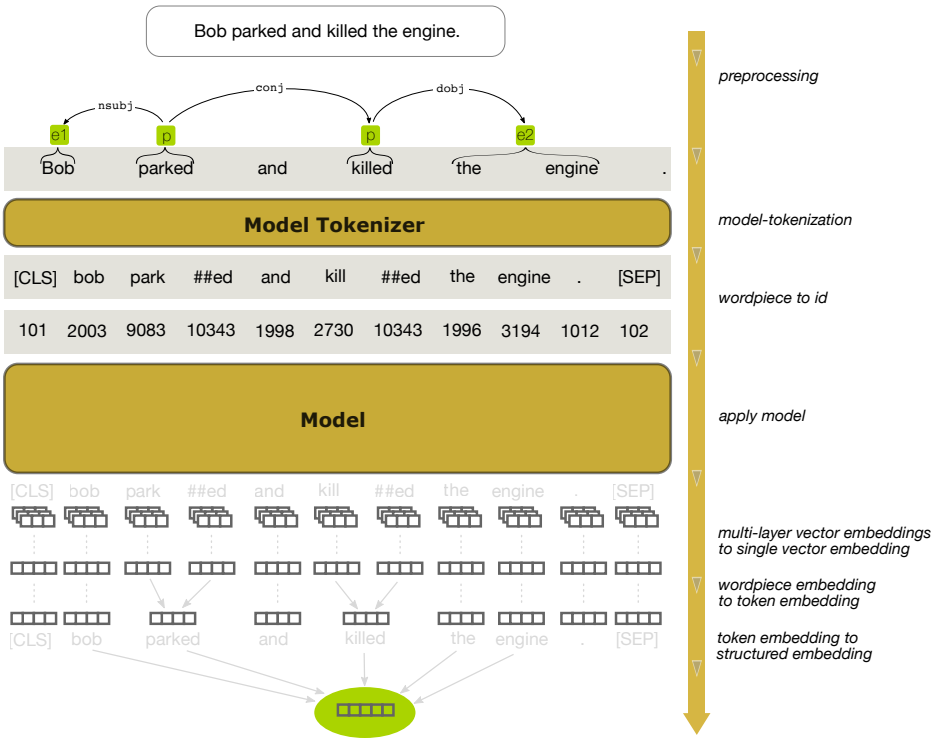


Figure 4.2: Structured embedding extraction process using linguistic patterns. In this example the `dep-path` pattern is illustrated.

4.4 Linguistically-Informed Patterns

We define various linguistic structure extraction patterns, which we test for their suitability to retrieve indexed sequences that are supposed to be semantically similar to a query sequence and its encoded semantic relation. Most of the patterns which we define in this section are general purpose patterns, i.e., they can be used to retrieve similar sequences without special treatment or extra information. For pre-processing, i.e., tokenization, part-of-speech tagging, and dependency parsing we use *spaCy*[1], and for chunking we use *flairNLP*[2]. To obtain an embedding vector for a token using CWEs based on RoBERTa (Robustly optimized BERT approach; Y Liu et al., 2019) wordpiece tokenization, we sum the output of the last four attention layers of the model and average all wordpieces of the particular token. Each of the following patterns, which involve multiple tokens, comes with an aggregation strategy to obtain the final sructured embedding. The entire process is illustrated in Figure 4.2. We use the following linguistically-informed patterns to find similar sentences with respect to the objective of semantic relation classification:

`token`: Each token of a sentence is considered to be a unique structure.

`word`: Same as `token` where punctuation is omitted.

`word-NS`: Same as `word` where stop-words are omitted.

1. https://spacy.io/
2. https://github.com/flairNLP/flair

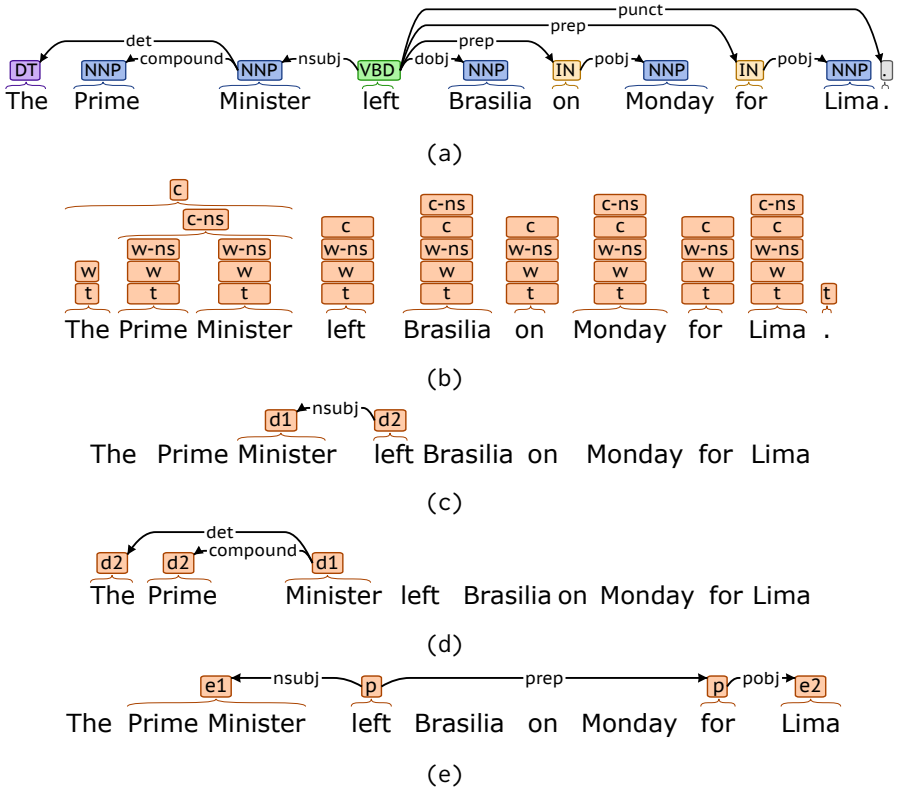


Figure 4.3: (a) shows the automatically extracted dependency graph and syntax features. (b-e) Linguistic patterns: (b) shows the aggregation strategy for `token` (t), `word` (w), `word-NS` (w-ns), `chunk` (c), and `chunk-NS` (c-ns). (c) shows the aggregation strategy for `dep-{concat,avg}` for a single dependency edge, i.e., $d1$ and its dependency head $d2$. (d) illustrates the `dep-depavg` strategy for the word 'Minister', where $d1$ is the actual word and all $d2$ are dependents of $d1$. (e) shows the task dependent `dependency-path` pattern for relation classification.

chunk: Each chunk of a sentence is considered to be a unique structure, where constituent token embeddings within a chunk are averaged.

chunk-NS: Same as chunk, but stop-words are omitted.

dep: Dependency relations are encoded as a combined vector of the relation's *head word* and its *tail word*. We test three strategies to encode the dependency relations as structured embeddings: *a*) both embeddings are concatenated (**-concat**) *b*) both embeddings are averaged (**-avg**) *c*) for each token of the sentence, we concatenate its embedding with the averaged embeddings of its dependent tokens, i.e., its tail words (**-depavg**). If a word does not depend on other words, i.e., it has no tail words attached, we concatenate an empty vector of the same size, i.e., a vector where all values are set to zero.

dep-path: The shortest dependency path between two given entities has been proven to be a beneficial feature for relation extraction in previous works (cf. for example among others: Lin and Pantel, 2001; Bunescu and Mooney, 2005; **Remus**, 2014). For this purpose, we define the structured embedding $\mathbf{x}$ of a sample to be the concatenation of the vectors for each entity $\mathbf{e}_{\{1,2\}}$ and the path $\mathbf{p}$: $\mathbf{x} := \mathbf{e}_1 \oplus \mathbf{e}_2 \oplus \mathbf{p}$. , Each individual vector $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{p})$ is the average of

all constituent word vectors. This pattern can be considered task-dependent since it requires the two entities to be given in advance. However, we want to stress that entities could be extracted in a NER pre-processing step and that each permutation of an entity pair could be used as a structure.

Figure 4.3 (b-e) shows the different structures for an example sentence, where Figure 4.3c only shows a single dependency relation as an example and is repeatedly applied for all dependency relations as exemplified in Figure 4.3a, and Figure 4.3d only shows a single example structure of the word '`Minister`' and is repeatedly applied for all words. We compare the sentence-level structures with the following two baseline approaches, which produce a single structured embedding for the entire sentence:

CLS: We use the artificial `[CLS]` token of BERT-based models (or the equivalent for non-BERT-based models such as RoBERTa) that is added to every sentence as a meta-token and which is often used as a vector representation for the entire sequence in downstream tasks.

BoW: All embeddings of all tokens are averaged to form a single embedding (bag-of-words).

4.5 Experiments

We investigate the retrieval performance using precision at k (P@k) and mean average precision (mAP) metrics. We use annotated datasets to evaluate the retrieval performance based on gold standard data, which means each sentence is labeled with one specific target class. The standard train and test splits are used for indexing and querying as indicated by each dataset. We also run a simple classification benchmark test using the same datasets for comparison purposes. As a classification approach, we opted to use a k-nearest neighbor (kNN) approach, which heavily relies on the retrieval performance, and thus, we implicitly evaluate the retrieval performance too. The kNN method groups and counts the class labels of the top-k retrieved training samples and uses the most prominent class label as a classification result. In case of ties, a random label of the most prominent classes is chosen. We report F_1 scores on the test sets and determine the hyperparameter k by using the validation set of the respective task benchmarks. If an explicit validation set is not supplied, we use an 80/20 split of the original training set and use a random subset for validation and training.

The identification of semantic relations is typically a multi-class classification problem, where our chosen datasets contain between 10 and 19 classes. We use three standard benchmarks from the SemEval[3] challenges for semantic relation classification: SE'07 (SemEval 2007; Girju et al., 2007), SE'10 (SemEval 2010; Hendrickx et al., 2010) and SE'18 (SemEval 2018; Gábor et al., 2018). Girju et al. (2007) and Hendrickx et al. (2010) focus on the classification of semantic relations between pairs of nominals. e.g., 'tea' and 'ginseng' are in an `ENTITY-ORIGIN(`e_1`,`e_2`)` relationship in the sentence 'The cup contained tea from dried ginseng.'. Gábor et al. (2018) focuses on domain-specific semantic relations from scientific articles

3. https://semeval.github.io/

Table 4.1: Results for the semantic retrieval of semantic relations. mAP refers to mean average precision, P@k refers to precision at k, w2v refers to the static word2Vec embedding and RB refers to the contextualized RoBERTa embedding.

	Pattern											
Data	CLS	BoW	token	word	word-NS	chunk	chunk-NS	dep-cat	dep-avg	dep-depavg	dep-path	
SE'18 (w2v)	-	32.9	31.1	30.4	31.2	31.0	30.9	30.6	30.8	31.2	***36.8***	mAP
	-	39.1	33.1	29.7	30.3	30.6	31.4	25.7	27.7	31.7	*46.0*	P@1
	-	34.6	30.9	31.7	33.4	31.4	31.3	27.1	30.2	31.1	*43.3*	P@5
SE'18 (RB)	31.9	34.5	32.1	31.4	32.0	32.1	32.2	31.8	32.2	32.4	*35.3*	mAP
	35.4	40.3	29.7	32.0	34.9	37.7	32.9	33.7	32.9	34.9	***52.9***	P@1
	34.6	37.8	33.5	32.2	35.0	33.4	34.9	35.0	33.4	34.1	***46.9***	P@5
SE'10 (w2v)	-	12.7	9.0	9.5	9.8	10.8	10.6	11.1	11.3	11.4	*22.2*	mAP
	-	35.5	9.9	14.4	15.6	21.9	21.8	22.7	22.5	23.0	*58.6*	P@1
	-	30.3	10.0	11.3	14.6	19.3	19.2	19.7	19.9	20.4	*50.0*	P@5
SE'10 (RB)	10.3	14.1	11.5	12.6	12.3	12.8	13.2	15.1	13.5	15.5	***26.5***	mAP
	31.6	40.6	26.0	26.8	27.3	32.0	32.4	38.3	27.5	37.6	***73.0***	P@1
	27.0	35.9	22.0	23.3	23.5	28.6	29.0	34.0	26.3	33.4	***66.5***	P@5
SE'07 (w2v)	-	32.2	29.2	29.6	29.8	30.5	30.4	30.5	30.6	30.8	***37.9***	mAP
	-	39.2	17.9	15.1	32.8	31.9	33.2	37.0	35.2	34.8	*53.6*	P@1
	-	36.5	20.2	22.9	30.2	32.2	32.4	31.8	32.6	33.3	*49.3*	P@5
SE'07 (RB)	30.8	31.6	30.6	31.2	31.5	31.1	31.3	32.2	31.3	32.5	*37.0*	mAP
	36.8	39.9	36.2	37.7	40.4	40.8	39.5	43.2	34.8	43.7	***61.9***	P@1
	33.7	37.3	32.9	34.9	35.6	37.2	35.3	39.9	34.3	38.6	***53.8***	P@5

and provides entire paragraphs instead of single sentences. In our information retrieval setup, we attempt to find semantically similar relation instances (sentences) from the training sets for every test set instance of the respective dataset.

Semantic Retrieval Results:
The results in Table 4.1 show that using RoBERTa in comparison to Word2Vec is beneficial, i.e., contextualized embeddings are more valuable than static word embeddings, however, the benefit is rather marginal, which is an interesting finding given that contextualized embeddings need way more data and computational resources for training. A common observation for all datasets is that simple linguistic patterns do not necessarily perform better in terms of small P@k values than the baseline `BoW` approach. Among the generally applicable linguistic patterns, the `dependency-depavg` performs consistently better than other patterns. Also, the baseline `BoW` approach consistently produces better results than the baseline `CLS` approach, which questions the practical usability of the `[CLS]` meta-token for downstream tasks. The specialized `dependency-path` pattern, however, improves the results by a large margin, almost doubling the `BoW` results and even tripling the `token` based results (cf. P@1, SE'10, RB).

Table 4.2: Concise classification results using kNN. Only the best settings and their scores are shown here, and complete results can be found in Table 4.3.

Data	Embedding	Masking	k	F_1
SE'07	w2v	dep-path	1	43.4
SE'10	RB	dep-path	5	78.7
SE'18	RB	dep-path	5	35.9

Table 4.3: Classification results using kNN showing the best identified hyperparameter k and the F_1 score.

	Pattern											
Data	CLS	BoW	token	word	word-NS	chunk	chunk-NS	dep-cat	dep-avg	dep-depavg	dep-path	
SE'18	10	4	14	15	18	38	27	25	2	20	5	k
(RB)	21.5	26.6	21.6	17.3	20.2	15.2	16.5	24.1	25.0	17.6	35.9	F1
SE'10	-	65	11	24	16	41	30	23	24	22	107	k
(w2v)	-	40.7	9.6	11.3	17.2	22.7	22.4	24.4	27.8	24.4	67.0	F1
SE'10	14	17	90	28	28	42	49	38	15	9	5	k
(RB)	33.9	50.5	24.8	29.0	30.0	34.8	34.1	31.2	40.2	41.7	78.7	F1
SE'07	-	1	16	6	10	2	7	2	1	16	1	k
(w2v)	-	22.3	8.6	7.0	8.6	12.6	12.6	14.7	16.5	11.3	43.4	F1
SE'07	6	2	5	4	10	3	3	11	3	1	12	k
(RB)	11.0	22.4	13.4	13.4	10.8	18.8	14.4	17.2	23.0	26.5	41.6	F1

kNN Classification Results

The results from our kNN classification experiments are concisely summarized in Table 4.2, where we show only the best-performing setup identified using a held-out validation set and tested on the held-out test set. Interestingly, the best classification setups correlate with the MaP scores from the retrieval setup, not the precision$k$ scores as one would intuitively expect. While the classification results are far from beating the state of the art, they confirm our hypothesis that contextualized embeddings encode more valuable semantic information than static word embeddings since they perform consistently better (except on the SE'07 task). For completeness, the list of all results can be found in Table 4.3.

kNN Classification Results w/ Increasing Data

In a further experiment, we measure the classifier's performance regarding the amount of data necessary to produce usable results. We randomly select subsets of the training data with an increasing amount of sentences and plot the results in Figure 4.4. We only show the best setups that were identified in Table 4.2. For SE'07 and SE'10, we can see that the available training data seems insufficient since the performance gradient is still high when reaching the maximum available training data. For SE'10, we can see that as few as $1,000$ samples are sufficient to reach a decent performance as compared to using the entire training data.

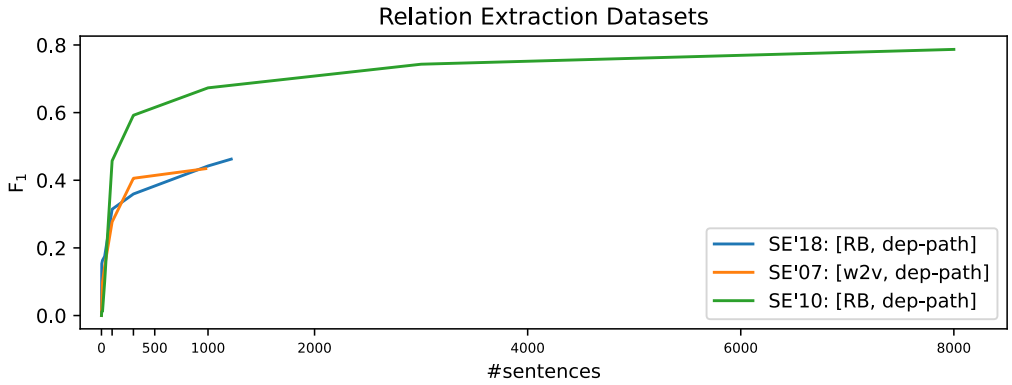


Figure 4.4: kNN performance for increasing training dataset sizes for relation classification. Only the best settings and their scores are shown.

4.6 Conclusion

We presented an analysis of different linguistically informed aggregation strategies for word embeddings in an information retrieval setting to find semantic units of the same class for different relation extraction datasets. For this, we used contextual word embeddings and linguistic structures as a means for indexing words or sentences and their class labels. We showed that the rather complex tasks benefit from both, linguistic structures and contextualized word embeddings. We also showed that for simple k nearest neighbor classification, only a certain amount of training data is sufficient to reach a decent performance. Use cases of this work include support for rapid training data collection, manual coding/annotation of datasets, e.g., in social science and humanities applications, retrieval of similar language use in eDiscovery tasks, and many more. In (**Remus** et al., 2022), we even extended this line of research and the experimental setup to more NLP tasks and showed that the benefit depends on the granularity of the task, i.e., the number of class labels. With finer-grained tasks, i.e., more class labels, the benefit of more advanced and task-dependent linguistic structures, such as the dependency path for relation extraction, becomes more apparent, whereas for simpler tasks, such as offensive language classification, which is a two-class classification problem, more superficial structures such as `token` or `chunk` perform best.

This chapter is based on the following published work by the author and collaborators:

Steffen Remus. 2014. Unsupervised Relation Extraction of In-Domain Data from Focused Crawls. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics,* 11–20. Gothenburg, Sweden.

Steffen Remus, Gregor Wiedemann, Saba Anwar, Fynn Petersen-Frey, Seid Muhie Yimam, and Chris Biemann. 2022. More Like This: Semantic Retrieval with Linguistic Information. In *Proceedings of the 18th Conference on Natural Language Processing (KONVENS 2022),* 156–166. Potsdam, Germany.

5 Unsupervised Semantic Relation Extraction

In this chapter, we present methodologies for clustering contextualized word embeddings (CWEs) with the goal of generating groups of words that are used to evoke similar semantic relations. We focus on verbs in this chapter because of their predicate-like authority in semantic relations, i.e., verbs are considered the main contributors for predicate-argument structured semantic relations (Nastase et al., 2013). Our motivation is *a*) to distinguish between the different senses a word can have in certain contexts, i.e., split polysemic word instances, and *b*) to find words and contexts that describe the same sense, i.e., merge synonymic words. A naïve approach to this could be to directly cluster all word instances within a corpus of sufficient size, such that it contains multiple occurrences of words expressing several senses. This, however, leads to severe resource issues because, even for small corpora and only verb instances, the list of contextualized embeddings for all words becomes too large to handle on a single computer. We parallelize this process by conceptually addressing polysemy first, i.e., we collect all sentences in which a particular word is used, cluster these instances into a manageable number of clusters per word, and run a secondary joint clustering on all word clusters as single instances, which, conceptually, addresses synonymy by merging clusters. Although the process is slightly different, we follow Ustalov et al. (2019) and call this process *local-global clustering*.

5.1 Related Work

Ustalov et al. (2019) presented Watset, a meta-algorithm that essentially performs fuzzy clustering of words in two hard clustering stages, i.e., by first resolving sense ambiguities of a word by using a symbolic neighborhood graph as input, and creating a so-called *intermediate sense graph* using vector comparison for further disambiguation of interrelated words. In more detail, a similarity graph of words is used as the input representation, although Ustalov et al. (2019) stress that the

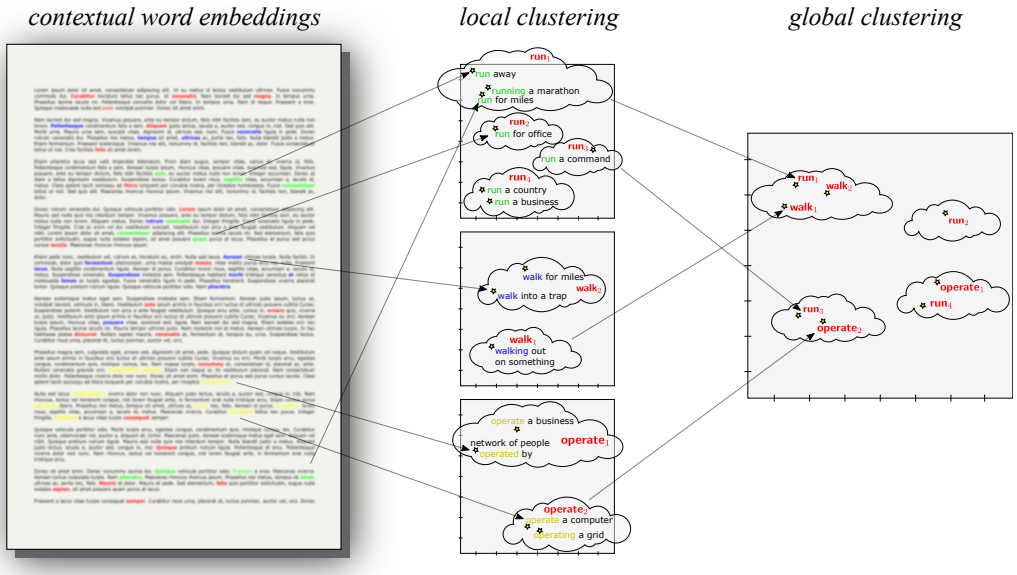


Figure 5.1: Schematic overview of the local-global clustering procedure.

approach itself is independent of words and just needs a similarity graph of any kind of object, they use word similarities in their experiments. Each subgraph of similarities that spans a particular word is then clustered into different senses, such that each neighbor of a word is assigned to a particular cluster. This procedure – called the *local step* – closely follows Dorow and Widdows (2003); more details can be found in (Ustalov et al., 2019; pp.10). For the so-called *global step*, the local cluster elements are disambiguated in order to form a global graph of senses instead of words. For this, the local clusters are aligned by maximizing a similarity measure, specifically the cosine similarity of adjacency encoded vectors (Ustalov et al., 2019, pp.13). In contrast to Ustalov et al. (2019), we do not rely on the disambiguation step of sense-related terms since ambiguities are modeled by the contextualized embeddings already (cf. Chapter 3.2). The globally disambiguated graph of senses is then clustered to create groups of senses, whereas the sense labels are removed in order to form proper synsets, i.e., sets of synonyms. The WATSET meta-algorithm is designed to be independent of the type of input nodes and the specific clustering algorithm. Hence, the clustering algorithms, and their respective parameterizations, are included as hyperparameters. We follow this setup and test the three following clustering methods *Chinese Whispers* (CW; Biemann, 2006), *Markov Clustering* (MCL; van Dongen, 2000), and *MaxMax* (MM; Hope and Keller, 2013).

5.2 Local-Global Clustering of CWEs

Given a sufficiently large corpus, our proposed procedure is as follows:

1. Collect all contexts x_i that contain a particular verb v from a defined set of verb types V.

$$X_v = \{ x_1, ..., x_n \mid v \in x_i \ \wedge \ v \in V \} \tag{5.1}$$

2. Cluster X_v such that v is split into n_v different senses while keeping a reference to the original context and its source sentence. We do this in order to provide provenance for analysis and explainability. The explicit clustering algorithm might create hard- or soft clusterings, i.e., clusters are allowed to have overlapping elements, but they do not necessarily have to overlap.

$$\text{cluster}_{\text{local}}(v) := \{ c_1^v, \dots, c_{n_v}^v \mid c_i^v \subseteq X_v \} \tag{5.2}$$

$$L = \bigcup_{\forall v \in V} \text{cluster}_{\text{local}}(v) \tag{5.3}$$

$$\tag{5.4}$$

   L is then the combined set of all local clusters.

3. Cluster L such that the local clusters are organized into m global clusters while still keeping a reference to a local cluster and the source contexts of its elements. This continues to serve provenance. The specific clustering algorithm might be different from the local clustering algorithm and might again create hard- or soft clusterings.

$$\text{cluster}_{\text{global}}(L) := \{ c_1^g, \dots, c_m^g \mid c_i^g \subseteq L \} \tag{5.5}$$

$$G = \text{cluster}_{\text{local}}(L) \tag{5.6}$$

   G is then the set of all global clusters, i.e., the final clustering.

The procedure is based on a *split-and-merge* idea, i.e., we split polysemic verb tokens of a particular verb type (local-step) and merge synonymic verb usages across all verb types (global step), see Figure 5.1 for an illustration of our procedure. Each step has its own design choices and parameterizations, which will be described in more detail in the following subsections.

5.2.1 Collecting Contexts (1)

We explore different contextualized vector representation extraction strategies for a particular verb v. Based on our previous findings in Chapter 4, where we have shown that contextualized embeddings, in combination with aggregation strategies based on linguistic knowledge, are helpful in building more discriminative representations of sentences, phrases, or words with special regard to a particular task (**Remus** et al., 2022; Chapter 3 and 4). Inspired by our experiments in Chapter 4, and the linguistic patterns we used for information retrieval; we investigate the effect of linguistic patterns as a masking strategy in the self-attention layers (Bahdanau et al., 2015) of transformer architectures (Vaswani et al., 2017). This gives us a linguistically-informed contextualized word embedding (CWE), which is internally estimated instead of externally aggregated as has been done in Chapter 4. While we averaged the contextualized embeddings based on a linguistic pattern in Chapter 4, we deviate from this strategy and utilize the internal attention mechanism of transformers to force the model to attend only to tokens that match a linguistic pattern. Below, we list the linguistic patterns we examined in this chapter:

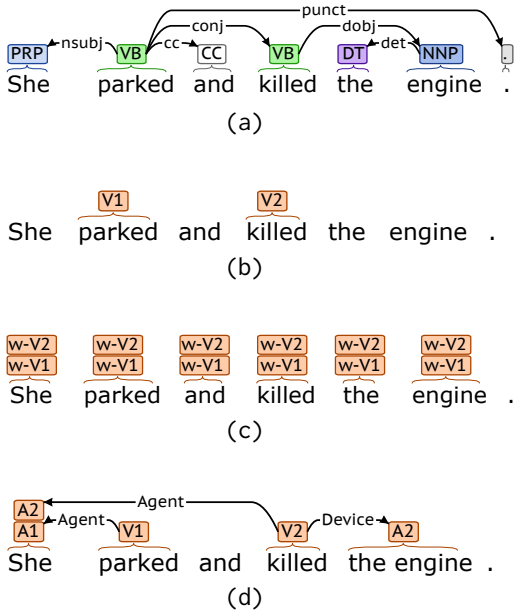


Figure 5.2: Linguistic patterns used for masking verb instances using attention. (a) shows the dependency graph for reference; (b) shows two verbs that are going to be used as the verb of interest by the patterns; (c) shows the non-stop-words which are used for attention; (d) shows the semantic roles provided by FrameNet annotations (`srl-*`).

`all`: default self-attention mask, where every word piece and special token has access to every other word-piece or special token,

`allw`: a self-attention mask, where every proper word, i.e., excluding special tokens, punctuation, etc., has access to every other proper word,

`verb`: only the verb of interest v has access to all other words,

`srl-args`: v has access to its arguments, but not to itself; additionally the arguments have access to themselves,

`srl-argsv`: v has access to its arguments and to itself; additionally the arguments have access to themselves and the verb v.

See Figure 5.2 for an illustration of the patterns and Figure 5.3 for a sketch of the collection procedure. For the **`srl-*`** strategies we use semantic role labeling (SRL) annotations provided by FrameNet[1] (Baker et al., 1998). We use the FrameNet annotations here for convenience because our experiments are based on the FrameNet corpus (cf. Section 5.4.1). Automatic SRL parser exist, which can be employed if no such annotations are at hand.

We also experiment with two different versions of input provisioning, i.e., the estimation of the structured embedding, for the **`srl-*`** patterns. First, we run the original sentence through the respective CWE model and use the attention masks for each verb in the sentence. For the second method, we gather all tokens of a frame, i.e., all its roles including the verb, and build a pseudo sentence from a particular pattern, which we then treat as a normal sentence to run through the

1. FrameNet version 1.7: https://framenet.icsi.berkeley.edu

CWE model for further processing and representation. Note that a single sentence might result in more than one pseudo sentence. The structure is used before the extraction process, whereas in the original extraction process, the structure is used in the embedding aggregation step. The incentive behind the latter strategy is to avoid noisy input from longer, possibly nested, sentences, with the possible pitfall of error propagation due to automatic parsing errors and incomplete sentences. Henceforth, the pseudo sentence strategy is marked with the subscript *ps* and the Figures 5.3a and 5.3b illustrate the two strategies.

Rogers et al. (2020) provide an extensive overview of the internal analyses of BERT (Devlin et al., 2019) and BERT-related models. Tenney et al. (2019) have shown, that different self-attention layers of the respective model behave differently for certain tasks. For example, while semantic properties are supposed to be spread across multiple layers, the first layers of a model are considered to be responsible for low-level syntactic properties of language such as parts-of-speech, chunks, etc., while subsequent layers are supposed to be responsible for higher level syntactical or grammatical properties, such as dependency parses or semantic roles, and the final layer is known to be most representative for task-specific semantic properties. Thus, the final layers often perform worse than medium range to penultimate layers for semantic clustering (NF Liu et al., 2019). Since we are interested in semantic clustering, we introduce the selection of the layer as a hyperparameter, where we restrict the choice to any permutation of the final three layers (i.e., [0,1,2], [0,1], [1,2], [0,2], [0], [1], [2], where 2 is the final layer, 1 is the penultimate layer, and 0 is the layer before that) and *concatenate* **or** *average* the respective layers.

Devlin et al. (2019) suggested using only some wordpiece embeddings mapped by a token. We follow their suggestion and the commonly applied policy but also test the most common strategies, such as:

`first` use only the first wordpiece of a token,

`last` use only the last wordpiece of a token,

`avg` average the embeddings of each wordpiece of a token.

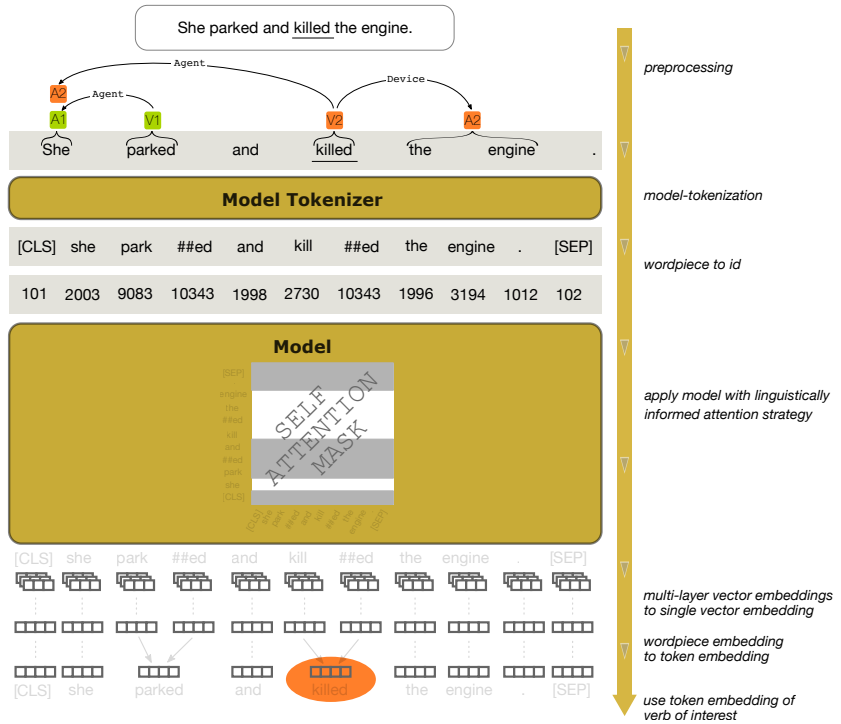
The entire extraction process is illustrated in Figure 5.3. We collect all verbs within a corpus and build a matrix X_v for each verb type v.

5.2.2 Locally Clustering Verbs in Context (2)

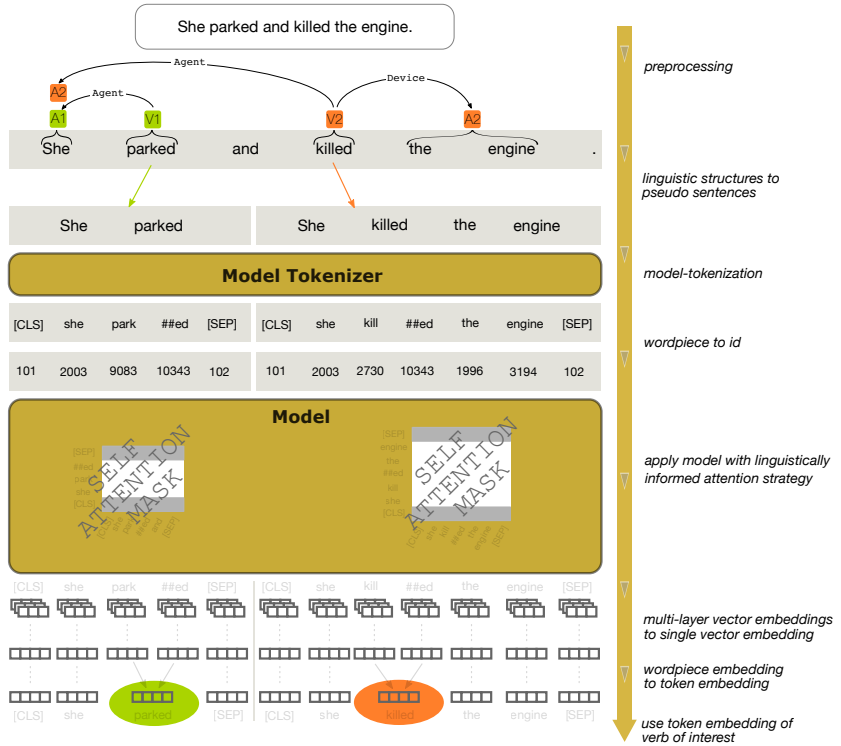
For a specific verb type v and its collection of contextualized word embeddings, we begin locally clustering the individual instances by *feature transformation*, i.e., the embeddings are transformed to a sparse vector representation of interpretable similarities. Here, we use cosine similarity and define different techniques for thresholding:

$$f(x) : \mathbb{R}^d \rightarrow \mathbb{R}^n \tag{5.7}$$

The matrix $X_v : \mathbb{R}^{n \times d}$ is transformed to $M_v : \mathbb{R}^{n \times n}$, where d is the embedding dimension, n is the number of instantiations of v, and an entry $m_{ij} \in M_v = \text{sim}(x_i, x_j)$ represents the similarity value between the embedding x_i and x_j. M_v can be interpreted as an adjacency matrix where each row i and column j represents a



(a)



(b)

Figure 5.3: Contextualized word embedding extraction process using linguistic patterns. (a) shows the default process for a particular verb v, in the example 'killed'; (b) shows the special case pseudo sentences construction before applying the model.

node and the similarity defines the weight of an edge between i and j. Pruning the matrix, by removing edges whose weights are below a specified threshold value, makes the graph sparse and transforms it into a so-called *small-world network* (Milgram, 1967), which can then be used for graph clustering. In a small-world network, nodes are defined to have only a few neighbors. We use the following methodologies for pruning edges from the initially fully connected graph. Pruning edges between nodes i and j in the adjacency matrix is done by setting the value $m_{ij} = 0$.

static$_\beta$: delete all edges whose weight is below a given static threshold β.

auto$_{mean}$: delete all edges whose weight is below the average value of M_v,

auto$_{median}$: delete all edges whose weight is below the median value of M_v,

auto$_{mean0}$: delete all edges whose weight is below the average value of M_v, where only values $m_{ij} > 0$ are used for computing the mean value,

auto$_{median0}$: delete all edges whose weight is below the median value of M_v, where $m_{ij} > 0$, where only values $m_{ij} > 0$ are used for computing the median value,

top$_k$: for each node, keep the top k (outgoing) edges, where symmetry is forced by adding (incoming) edges since the similarity graph is bi-directional.

cc$_\beta$: iteratively delete the lowest valued edge while the estimated *clustering coefficient* is below a certain threshold β. Since we want the graph to be symmetric, we always delete both edges between nodes i and j, i.e., $m_{ij} = m_{ji} = 0$.

Additionally, we delete self-loops ($m_{ii} = 0$) and optionally binarize edge weights that are greater than zero after pruning, i.e.,

$$m_{ij} = \begin{cases} 1 & \text{iff } m_{ij} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

Graph clustering algorithms exploit the small-world network property and try to find clusters of nodes that are highly connected, i.e., nodes that have higher *intra-connections* than *inter-connections* across other clusters. We opted for algorithms with a small set of hyperparameters for the clustering procedure. We argue that especially the number of clusters should never be set as a hyperparameter because truly unsupervised learning algorithms will find differences between samples that are not obvious to humans and do not necessarily correlate to human-assigned categories or even the number of categories. The following clustering algorithms fulfill our chosen criteria:

mcl: Markov Clustering (van Dongen, 2000) implements a random walker paradigm similar to the *page-rank* algorithm (Page et al., 1999).[2]

2. We stick to the recommended default parameters since we did not perceive any substantial changes when adjusting them.

mm: MaxMax (Hope and Keller, 2013) is a parameterless soft graph-clustering algorithm, which means that nodes might be assigned to one or more clusters with a certain probability. The algorithm transforms any undirected graph into a directed graph and finds longest paths, where each path is going to be a cluster, and each node on a path is a cluster element.

cw$_{\{max,sl,sn\}}$: Chinese Whispers (Biemann, 2006) is an efficient graph clustering algorithm based on the idea of *label propagation* and comes with a tiny set of hyperparameters.

Spectral Clustering (Shi and Malik, 2000), *Affinity Propagation* (Frey and Dueck, 2007), and *DBSCAN* (Ester et al., 1996) have been tested in preliminary experiments with the conclusion that they mainly fail to scale to large amounts of data samples.[3] For Chinese Whispers, we define some novel *label propagation* variants (see Section 5.3), where we use the current task for evaluating their performance. We thus have the following variations for Chinese Whispers:

cw$_{max}$: specifies the default maximum class label strategy of Chinese Whispers.

cw$_{sl}$: here, we use the sampling method for class labels propagated from a node's neighbors as described in Section 5.3.

cw$_{sn}$: here, we use the sampling method for neighbors as described in Section 5.3.

We stick to the recommended default parameter set for weighting individual nodes.

Clustering the localized samples, i.e., the verbs in their defined context, is then straightforward using the specific clustering algorithm. The result is a local clustering for each verb type v, where we define L to be the union of all clusters across all verbs, i.e., L is the joint set of all local clusters. In the local clustering step, we conceptually dissolve the polysemous verb type v and group ambiguous instantiations of v into clusters in L.

5.2.3 Globally Clustering Local Clusters (3)

The global clustering step groups the clusters from the local clustering step across verb types. The main idea behind this step is to resolve synonymous expressions or clusters. For this, we examine the same clustering methods as we did for the local clustering step: **mcl**, **mm**, **cw**$_{\{max,sl,sn\}}$. To find a suitable vector representation for a local cluster, we define the following two heuristics:

average: the most straightforward approach is to average all the elements of a local cluster.

first: here, we rank the elements of a cluster and use the embedding representation of the first element as a representation for the entire cluster. This ranking heavily depends on the local clustering algorithm's ability to produce intra-cluster element rankings. MaxMax, for example, comes with an inbuilt ranking of clustered nodes due to using the longest path; here, we use the root node to represent the cluster. In Chinese Whispers and Markov

3. We used the implementations provided by *scikit-learn* https://scikit-learn.org/.

Clustering, we use the node with the highest degree and choose a random node in case of ties.

rand: here, a random node is chosen to represent the cluster.

Each local cluster in L is thus represented by a single vector; we collect all vectors and call the resulting matrix X_L and apply the same feature transformation steps as in the local clustering step, i.e., we *a*) transform X_L into an adjacency matrix M_L by computing pairwise similarities, and *b*) prune the graph such that it fulfills the small-world network property. We test the same options and techniques as listed above. Additionally, we remove edges between clusters generated from the same verb type. Once M_L is pruned, we apply the same techniques for clustering as described above, namely **mcl**, **mm**, or $\mathbf{cw}_{\{max,sl,sn\}}$.

5.3 Chinese Whispers with Sampling

Chinese Whispers (CW; Biemann, 2006) is a type of a label propagation algorithm within the family of graph clustering algorithms. Each node is initially assigned a unique class label and iteratively propagates its label to neighboring nodes. The label of a node is updated by choosing one of the class labels from the neighboring nodes, where the default strategy is to select the highest-ranked class label within the local neighborhood of a node. The rank of a class is computed as the sum of the weights of each neighboring node that is currently assigned the particular class label. Biemann (2007) introduced multiple scoring functions that consider the edge weight between the two nodes and the incoming and outgoing degree of the propagating node. The pseudo-code for the generic algorithm can be found in Listing 5.1 and Listing 5.2. Further details can also be found in (Biemann, 2006, 2007, 2012).

Since Chinese Whispers also constitutes properties of a random walker in a Markov process, we added two variants, which, instead of computing the arg max of neighboring labels, use sampling from a probability distribution. Consider the pseudo algorithm in Listing 5.2: in Lines 8-10, the weights of neighboring nodes are aggregated and form a local score for each label given a node v. The default procedure is then to use the label with the highest score for label propagation (Line 13). However, we argue that a probabilistic interpretation might help to bring more stability to CW's non-deterministic behavior. In Line 17, we introduce a sampling variant that treats the scores of labels as a local probability distribution $p(c|v)$ of a class c given a node v and draws a (label) sample from this distribution. We call this method $\mathbf{cw}_{sl}$ (sample label). In Line 21, we introduce a sampling variant that treats the weights of neighboring nodes as a local probability distribution $p(u|v)$ of a neighbor u given a node v. Similar to $\mathbf{cw}_{sn}$, the idea is to draw a node u from $p(u|v)$ and use its class label c_u as the new class label for v. We call this method $\mathbf{cw}_{sn}$ (sample neighbor). For both methods, the law of large numbers provides the foundation of our idea that the selected label should converge to the most probable label in its vicinity.

However, we note that in theory, $\mathbf{cw}_{sl}$ and $\mathbf{cw}_{sn}$ should yield the same result because the probability mass of a label is the sum of the probabilities of the neighboring vertices with that label. The difference is that $\mathbf{cw}_{sn}$ implicitly scatters

Listing 5.1: The generic Chinese Whispers (CW) algorithm. Vertices with the same class label build the final clusters.

```
// initialize class labels
for i = 1 to |V| do
  c_i ← i
end for
// convergence is reached if
//  - the maximum number of iterations is reached, or
//  - changes are neglectable
until convergence do
  // update class labels
  for each v ∈ shuffle(V) do
    c_v ← select_label(v)
  end for
end for
```

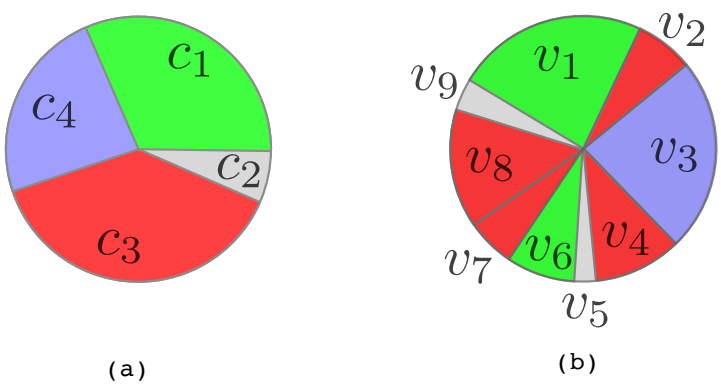


Figure 5.4: Illustration for the distribution of probability masses for class labels collected from neighboring nodes. The entire circle represents 100 percent. Colors and $c_1 \dots c_4$ refer to classes; $v_1 \dots v_9$ refer to neighboring nodes with the color being the assigned class. (a) shows the distribution of classes when aggregated over the neighbors; (b) shows the distribution of classes when not aggregated.

probabilities of labels whereas $\mathbf{cw}_{sl}$ collects it in a bulk (cf. Figure 5.4 for illustration). This difference, we believe, might have an influence in a practical setting; we thus treat and test both strategies independently.

5.4 Experiments

5.4.1 Dataset

Experiments are performed using FrameNet (Baker et al., 1998)[4]. We collect only sentences where the frame evoking lexical unit is a verb, i.e., each sentence contains at least one verb-frame. FrameNet provides multiple sentences per frame,

4. We use FrameNet version 1.7 provided by the NLTK Framework (Bird et al., 2009).

Listing 5.2: Label propagation (selection) strategies for Chinese Whispers (CW). Details of the 'weight(·, ·)' function, to compute the local weight of node, can be found in (Biemann, 2012).

```
function select_label
  input: node v
  // select a new label by the maximum weight of the neighbor's class labels
  let E_v^in be the set of weighted *incoming* edges of v
  let U_v^in be the set of neighboring nodes of v in E_v^in
  let C_v^in be the set of unique class labels of nodes in U_v^in
  // initialize aggregated class label scores
  for i = 1 to |C_v^in| do
    s_i ← Σ_{(u,v)∈E_v^in} weight(u,v) × c_u == c_i
  end for

  // OPTION: CW_max (default)
  c_v ← arg max(s)

  // OPTION: CW_sl
  // treat aggregated class scores as a probability distribution and sample from it
  c_v ← draw_sample_class(s)

  // OPTION: CW_sn
  // treat weights of nodes u ∈ U_v^in as a probability distribution and sample from it
  u ← draw_sample_node({ weight(u, v) ∀ u ∈ U_v^in })
  // get the class of sampled node
  c_v ← c_u
```

which might be evoked by one or more verb types. We can thus use it directly to evaluate polysemy and synonymy. We treat the annotated frame for a sentence, and the frame-evoking verb, as a gold cluster label, and the collection of all frames and sentences defines our gold clustering. Table 5.1 lists some statistics of the dataset. Figure 5.5 (b & c) show some statistics about the distribution of verb types across frames and their contextualized instances within the sentences. Figure 5.5a shows the distribution of sentences for the verb types. Figure 5.5b shows the distribution of frames per verb type, i.e., here, we can read the synonymic nature of frames in the FrameNet data. For example, we can see that roughly 175 frames are instantiated by only one verb type, i.e., there's no verbal synonymy present in the data, whereas a few frames are instantiated by 20 verbs or more; those frames can be considered highly synonymous. The polysemic nature of the data can be read in Figure 5.5c, which shows the distribution of verb types per frame. For instance, we can see that roughly 1,900 verb types instantiate only a single frame, i.e., in this case, there is no expected ambiguity of the verb. However, we can also see that more than 1,000 verb types instantiate two or more frames, and a few verb types even instantiate five or more frames; those verb types can be considered highly polysemic. We can observe that all distributions follow a power-law pattern, e.g., many frames are evoked by only a few verb types, and only a few frames are

Table 5.1: FrameNet statistics. Note that the numbers refer only to frame-evoking lexical units that are verbs. The number of frames implicitly defines the number of gold clusters. Sentences might invoke multiple frames (cf. the example in Figure 4.2).

#verb types	2,663	
#frames	642	*defines the target number of global clusters*
#verb-frames	3,877	*defines the target number of total local clusters*
#sentences	76,763	
#frame instances	82,623	
#tokens	1,868,156	
vocabulary size	77,624	

evoked by multiple verb types (cf. Figure 5.5b), this directly translates to resolving synonymy in our global clustering step. Also, in Figure 5.5c, we can see that many verb types can be associated with only a few frames, whereas only a few verb types can be associated with multiple frames, which translates to resolving polysemy in our local clustering step. Sentences are distributed in a power-law fashion too, i.e., many verb types occur in only a few sentences, whereas only a few verb types occur in many sentences (cf. Figure 5.5a).

5.4.2 Embedding Models

We use the following contextualized embedding models but note that those can be easily exchanged by other (contextualized) embedding models since the model only provides features for the procedure itself. The only constraint we set to the pre-trained model architecture is the internal use of the attention mechanism (Bahdanau et al., 2015) so that we can evaluate strategies regarding attention masks. For ease of use, we stick to models implementing variants of the transformer architecture first proposed by Vaswani et al. (2017).

BERT (Bidirectional Encoder Representations from Transformers) was introduced by Devlin et al. (2019) and is probably the first and most popular choice of the transformer architecture because at the time of publication, BERT achieved state-of-the-art results in many NLP downstream tasks. Its introduction also offered easy access to pre-trained models in different sizes, i.e., different numbers of trainable parameters, etc. Since its introduction, many successor architectures have been proposed.

RoBERTa (Robustly optimized BERT approach) is a BERT successor model by Y Liu et al. (2019), which uses a dynamic token masking and prediction strategy for pre-training, which enables the easier exchange of underlying dataset. BERT, on the other hand, applied the token masking during pre-processing per sentence, and thus, the model is always presented the same statically masked tokens when making a pass over the data.

SpanBERT is another pre-training method based on BERT's architecture proposed by Joshi et al. (2020). SpanBERT is designed to represent the prediction of

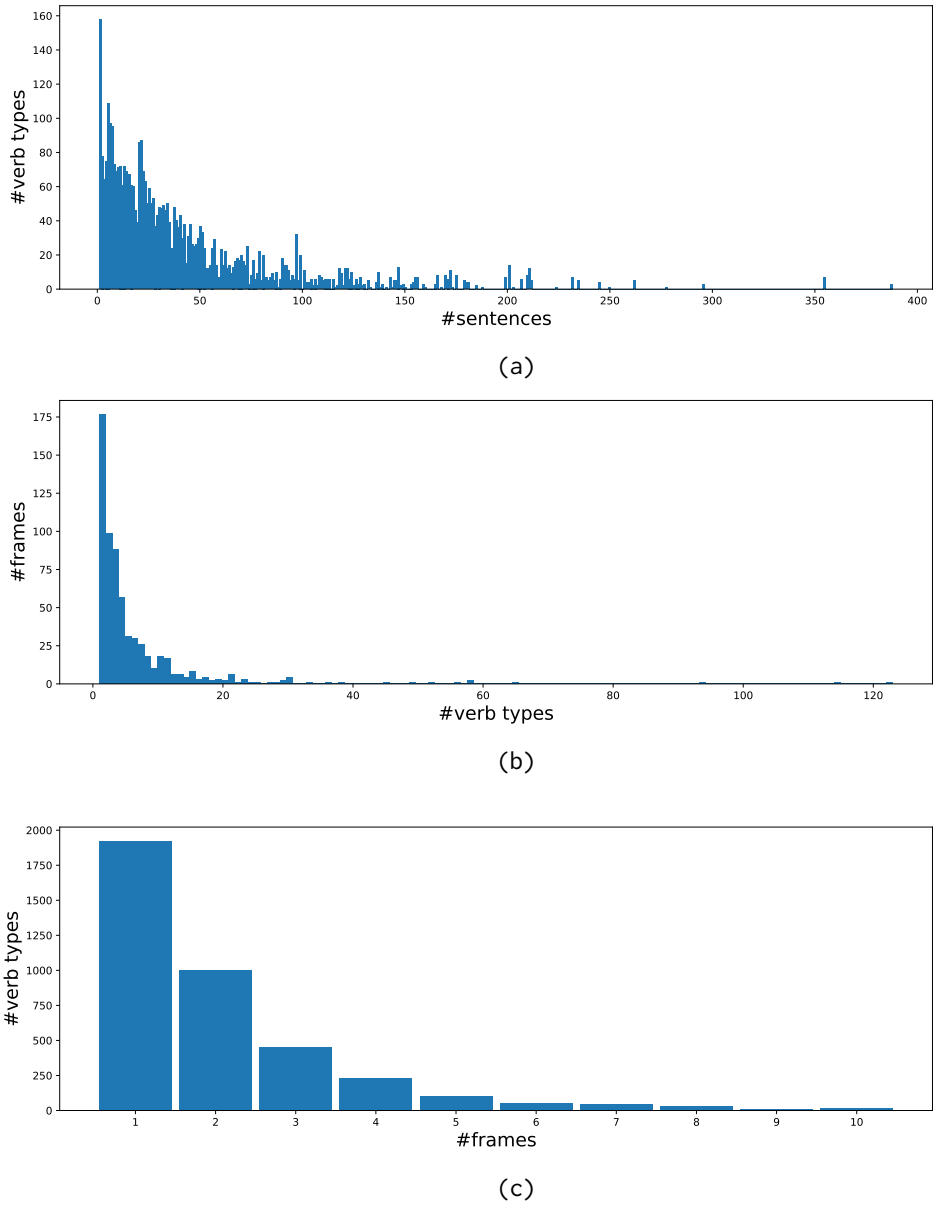


Figure 5.5: Histograms generated from the FrameNet where the lexical units invoking frames are verbs. (a) shows the distribution of the number of sentences per verb type; (b) shows the distribution of frames per verb type, i.e., here we can read the synonymic nature of the verb type data; and (c) shows the distribution of the number of frames a verb type can invoke, i.e., here we can read the polysemic nature of verbs.

text spans better than the individual and independent prediction of single tokens. Thus, a mask can represent either a phrase of multiple tokens or a single token, e.g., in the sentence "I love [MASK]", the special token "[MASK]" can be used to predict a single word like "Berlin" or a span like "the city of New York".

5.4.3 Clustering Evaluation Metrics

Many metrics exist for comparing gold clusterings with generated clusterings. In (**Remus**, 2012; **Remus** and Biemann, 2013), we explored several clustering

Table 5.2: Scores of the top 20 final global clusterings as defined by the combined relative ranking across all clusters.

model	*mask*	*token emb*	*emb layer*	*local clustering*	$r_{local}(\phi)$	*global clustering*	*local cluster repr*	*FMI*	*r(FMI)*	*AMI*	*r(AMI)*	PUF_1	$r(PUF_1)$	*NMI*	*r(NMI)*	*ARI*	*r(ARI)*	$\phi(r(\cdot))$
BERT	all	avg	[2]	$\texttt{cw}_{it}(\texttt{cc}_{.5})$	104	$\texttt{cw}_{max}(\texttt{auto}_{median})$	avg	.080	546	.621	3	.484	384	.681	268	.052	228	286
BERT	all	avg	[2]	$\texttt{cw}_{it}(\texttt{cc}_{.5})$	104	$\texttt{cw}_{max}(\texttt{auto}_{mean})$	avg	.080	545	.621	1	.484	387	.681	269	.052	227	286
BERT	all	avg	[2]	$\texttt{cw}_{it}(\texttt{cc}_{.5})$	104	mm	avg	.080	543	.621	2	.484	398	.681	271	.052	226	288
SpanBERT	all	avg	[1]	$\texttt{cw}_{it}(\texttt{cc}_{.5})$	77	$\texttt{cw}_{max}(\texttt{auto}_{median})$	avg	.093	98	.612	11	.453	779	.673	473	.075	125	297
SpanBERT	all	avg	[1]	$\texttt{cw}_{it}(\texttt{cc}_{.5})$	77	$\texttt{cw}_{max}(\texttt{auto}_{mean})$	avg	.093	98	.612	12	.453	779	.673	473	.075	125	297
SpanBERT	all	avg	[1]	$\texttt{cw}_{it}(\texttt{cc}_{.5})$	77	mm	avg	.093	98	.612	13	.453	779	.673	474	.075	125	298
SpanBERT	all	first	[1]	$\texttt{cw}_{it}(\texttt{cc}_{.5})$	76	$\texttt{cw}_{max}(\texttt{auto}_{median})$	avg	.091	127	.607	26	.448	853	.670	540	.074	126	334
SpanBERT	all	first	[1]	$\texttt{cw}_{it}(\texttt{cc}_{.5})$	76	$\texttt{cw}_{max}(\texttt{auto}_{mean})$	avg	.089	166	.608	20	.446	867	.672	494	.073	129	335
SpanBERT	all	first	[1]	$\texttt{cw}_{it}(\texttt{cc}_{.5})$	76	mm	avg	.089	165	.607	22	.446	876	.671	513	.073	128	341
BERT	all	avg	[1]	$\texttt{cw}_{it}(\texttt{cc}_{.5})$	117	$\texttt{cw}_{max}(\texttt{auto}_{mean})$	avg	.080	548	.613	8	.479	446	.671	506	.049	246	351
BERT	all	avg	[1]	$\texttt{cw}_{it}(\texttt{cc}_{.5})$	117	mm	avg	.080	538	.612	10	.478	453	.671	523	.050	244	354
BERT	all	avg	[1]	$\texttt{cw}_{it}(\texttt{cc}_{.5})$	117	$\texttt{cw}_{max}(\texttt{auto}_{median})$	avg	.079	585	.613	9	.479	445	.672	495	.049	255	358
BERT	all	avg	[2]	$\texttt{cw}_{in}(\texttt{cc}_{.5})$	129	$\texttt{cw}_{max}(\texttt{auto}_{mean})$	avg	.076	755	.611	14	.483	410	.673	455	.045	312	389
BERT	all	avg	[1,2]	$\texttt{cw}_{it}(\texttt{cc}_{.5})$	121	mm	avg	.077	745	.609	18	.479	439	.671	516	.047	281	400
BERT	all	avg	[2]	$\texttt{cw}_{in}(\texttt{cc}_{.5})$	129	$\texttt{cw}_{max}(\texttt{auto}_{median})$	avg	.076	793	.611	15	.481	431	.673	464	.044	324	405
BERT	all	avg	[2]	$\texttt{cw}_{in}(\texttt{cc}_{.5})$	129	mm	avg	.075	846	.610	16	.480	437	.673	478	.044	330	421
SpanBERT	all	avg	[1]	$\texttt{cw}_{in}(\texttt{cc}_{.5})$	115	$\texttt{cw}_{max}(\texttt{auto}_{mean})$	avg	.083	359	.602	36	.446	872	.665	688	.063	157	422
SpanBERT	all	first	[1]	$\texttt{cw}_{in}(\texttt{cc}_{.5})$	107	mm	avg	.085	300	.598	46	.442	936	.663	785	.066	145	442
BERT	all	avg	[1,2]	$\texttt{cw}_{it}(\texttt{cc}_{.5})$	121	$\texttt{cw}_{max}(\texttt{auto}_{mean})$	avg	.075	840	.607	25	.474	493	.669	565	.046	300	445
SpanBERT	all	first	[1]	$\texttt{cw}_{in}(\texttt{cc}_{.5})$	107	$\texttt{cw}_{max}(\texttt{auto}_{mean})$	avg	.085	295	.597	48	.441	955	.662	797	.066	144	448

comparison measures and confirmed previous findings that the perfect clustering evaluation metric does not exist. In this work, we use a combination of the following metrics: AMI (adjusted mutual information); FMI (Fowlkes-Mallows Index, also known as the pairwise F_1); PUF_1 (Purity F_1); NMI (normalized mutual information), which gives same score as the default V-measure, which is computed as the weighted harmonic mean of Homogeneity (H) and Completeness (C) (cf. **Remus**, 2012); and ARI (adjusted rank index).

Because each metric is not necessarily normalized or limited by an upper bound, simple averaging strategies cannot be applied, but since we compute many clusterings with different hyper-parameter setups, we can compute a relative rank for every clustering and metric. We denote the rank of a clustering regarding a metric with $r(metric)$. Using the ranks, we are able to aggregate all ranks of each metric to produce a single rank as our final score: $\phi(r(\cdot))$ denotes the average rank of all metrics and is used for sorting clusterings.

5.4.4 Evaluation Procedure

Using grid search, we compute all clusterings for several combinations of parameters listed in Section 5.2, which amounts to roughly 10K different clusterings. Since the cluster evaluation metrics do not necessarily correlate, we first compute the rankings for each of the five metrics and compute the average of the ranks, which provides an approximation of the agreement between the metrics w.r.t. their relative rank performance across different clusterings. The average rank can then be used as a meta-score for comparison purposes with a single evaluation score.

5.4.5 Results

Results of the entire local-global clustering procedure are listed in Table 5.2. We can see that the top hyper-parameter setups often share the same hyper-parameters from the local clustering step, i.e., the top entries come in triples, meaning they share the same local clustering setup and only differ in the global clustering method. Here, $\texttt{cw}_{max}$ method alternates with the $\texttt{auto}_{median}$ and $\texttt{auto}_{mean}$ pruning strategy. This seems intuitive since the $\texttt{auto}_{mean}$ and $\texttt{auto}_{median}$ pruning strategies remove a similar amount of graph edges, the difference is negligible, and they even produce very similar scores, such that the final average ranks are in fact

Table 5.3: Scores of the top 20 local clusterings as defined by the combined relative ranking across all clusters.

(a) Local clustering results.

id_{local}	model	mask	token emb	emb layer	local clustering	FMI	r(FMI)	AMI	r(AMI)	PUF_1	$r(PUF_1)$	NMI	r(NMI)	ARI	r(ARI)	$\phi(r(\cdot))$
1	RB	all_{ps}	avg	[1]	**mcl(top$_{10}$)**	.783	1	.926	1	.882	1	.965	1	.780	1	1
2	RB	all_{ps}	avg	[0,1,2]	**mcl(top$_{10}$)**	.782	2	.925	2	.881	2	.965	2	.779	2	2
3	RB	all_{ps}	first	[0,1,2]	**mcl(top$_{10}$)**	.782	3	.924	3	.879	3	.964	3	.778	3	3
4	RB	all_{ps}	avg	[2]	**mcl(top$_{10}$)**	.779	4	.923	4	.879	4	.964	4	.776	4	4
5	RB	all_{ps}	first	[1]	**mcl(top$_{10}$)**	.779	5	.923	5	.878	5	.964	5	.776	5	5
6	RB	all	avg	[1]	**mcl(top$_{10}$)**	.777	6	.922	6	.877	6	.964	6	.773	6	6
7	RB	all	avg	[0,1,2]	**mcl(top$_{10}$)**	.776	8	.922	7	.876	7	.964	7	.773	8	7
8	RB	all_{ps}	first	[2]	**mcl(top$_{10}$)**	.776	7	.921	8	.876	8	.963	8	.773	7	8
9	RB	all	avg	[2]	**mcl(top$_{10}$)**	.772	9	.919	10	.874	9	.962	10	.769	9	9
10	BERT	allw	avg	[2]	**mcl(top$_{10}$)**	.770	10	.920	9	.874	10	.963	9	.768	10	10
11	BERT	allw	avg	[0,1,2]	**mcl(top$_{10}$)**	.762	17	.917	11	.871	11	.962	11	.760	18	14
12	RB	all	first	[0,1,2]	**mcl(top$_{10}$)**	.762	16	.915	13	.869	12	.961	13	.760	17	14
13	SpanBERT	all_{ps}	avg	[2]	**mcl(top$_{10}$)**	.765	13	.914	15	.867	17	.960	15	.762	14	15
14	BERT	allw	first	[0,1,2]	**mcl(top$_{10}$)**	.760	21	.916	12	.869	13	.961	12	.758	20	16
15	RB	allw	avg	[2]	**mcl(top$_{10}$)**	.766	11	.913	17	.865	21	.960	17	.763	13	16
16	BERT	all	avg	[1]	**mcl(top$_{10}$)**	.765	12	.911	21	.867	16	.960	20	.764	11	16
17	BERT	all	avg	[2]	**mcl(top$_{10}$)**	.765	14	.911	20	.868	15	.960	19	.763	12	16
18	RB	all	first	[1]	**mcl(top$_{10}$)**	.760	22	.914	14	.868	14	.960	14	.758	22	17
19	SpanBERT	all_{ps}	avg	[0,1,2]	**mcl(top$_{10}$)**	.760	20	.913	18	.866	19	.960	18	.758	21	19
20	SpanBERT	all	avg	[0,1,2]	**mcl(top$_{10}$)**	.759	23	.912	19	.866	20	.959	21	.757	23	21

(b) Global clustering results by using only the top 20 local clusterings.

id_{local}	global clustering	local cluster repr	FMI	r(FMI)	AMI	r(AMI)	PUF_1	$r(PUF_1)$	NMI	r(NMI)	ARI	r(ARI)	$\phi(r(\cdot))$	$r(\phi)$
10	**cw$_{max}$(auto$_{mean}$)**	avg	.072	1062	.551	336	.432	1160	.608	1800	.028	911	1054	144
10	**cw$_{max}$(auto$_{median}$)**	avg	.073	1055	.550	346	.430	1190	.607	1811	.028	906	1062	146
10	**mm**	avg	.073	1016	.549	350	.429	1221	.605	1842	.028	892	1064	149
11	**mcl(top$_{10}$)**	avg	.058	2505	.548	361	.487	327	.677	359	.019	1990	1108	158
10	**mcl(top$_{10}$)**	avg	.057	2661	.553	319	.490	258	.681	282	.018	2060	1116	163
10	**mcl(cc$_{.5}$)**	avg	.057	2675	.554	317	.491	249	.681	283	.018	2071	1119	164
14	**mcl(cc$_{.5}$)**	avg	.059	2461	.544	394	.488	306	.675	430	.018	2049	1128	165
11	**mm**	avg	.074	906	.537	437	.416	1432	.592	2020	.029	857	1130	167
14	**mcl(top$_{10}$)**	avg	.058	2471	.544	396	.488	307	.675	433	.018	2058	1133	168
11	**mcl(cc$_{.5}$)**	avg	.058	2575	.548	363	.486	332	.677	363	.019	2043	1135	169
11	**cw$_{max}$(auto$_{mean}$)**	avg	.074	930	.536	446	.417	1423	.592	2014	.028	866	1136	170
11	**cw$_{max}$(auto$_{median}$)**	avg	.073	998	.537	441	.419	1392	.594	1998	.028	910	1148	173
20	**mcl(cc$_{.5}$)**	avg	.068	1556	.497	767	.513	37	.640	1336	.016	2544	1248	215
20	**mcl(top$_{10}$)**	avg	.068	1567	.496	777	.513	36	.640	1334	.016	2551	1253	219
15	**mcl(cc$_{.5}$)**	avg	.077	740	.462	1244	.519	11	.611	1771	.016	2579	1269	228
15	**mcl(top$_{10}$)**	avg	.077	736	.462	1245	.519	9	.611	1770	.016	2593	1271	230
5	**mcl(top$_{10}$)**	avg	.054	3024	.553	325	.492	228	.680	308	.016	2563	1290	237
13	**mcl(top$_{10}$)**	avg	.068	1536	.492	842	.512	38	.636	1401	.016	2702	1304	246
5	**mcl(cc$_{.5}$)**	avg	.054	3023	.552	333	.490	261	.678	340	.016	2568	1305	248
13	**mcl(cc$_{.5}$)**	avg	.067	1596	.491	846	.511	46	.634	1420	.015	2776	1337	256

the same. The third algorithm that performs in the same range is **mm**, which is surprising since it does not require explicit pruning of edges by a hyperparameter. As expected, it also becomes apparent that the metrics do not correlate, e.g., the top entries are ranked relatively high with respect to the AMI metric, whereas the values for the other metrics are in the higher range. Also, according to the ranks, the best local clusterings are not contributing to the best global clustering performance; their rank performance ranges from 76 to 129. The preferred global clustering mainly depends on the BERT model, the **all** attention mask across the entire source sentence, simple word piece averaging (**avg**), the final layer ([2]) or pen-ultimate layer ([1]), or the concatenation of both ([1,2]), and mainly on the Chinese Whispers with sampling algorithm (**cw**$_{sl}$ or **cw**$_{sn}$) with cluster coefficient threshold pruning (**cc**$_{.5}$). Only eight of all local clusterings are represented in the top 20 final clusterings.

Tables 5.3a and 5.3b show results for the individual steps, i.e., first, we present results for the local clustering step in Table 5.3a, where we split gold clusters by their lemmas for evaluation. Because of this, the total number of gold clusters amounts to 3, 877 (cf. Tab. 5.1) and are thus more fine-grained. Second, we present results of the global clustering step by using only the top 20 local clusterings

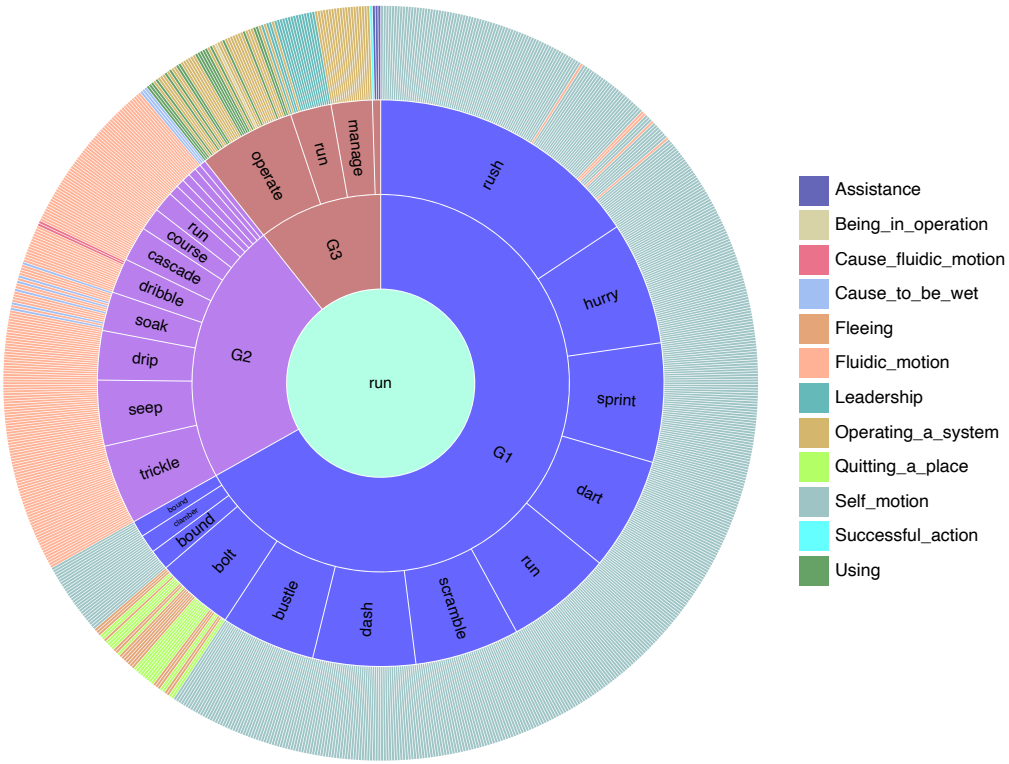


Figure 5.6: Final clusters where the word 'run' is involved and their gold labels as defined by FrameNet. The outermost circle represents the sentences in which the respective word of the local clusters (second outermost circle) occur. The legend covers the frame's name, evoked by the verb in the sentence (only outermost colors).

from Table 5.3a in Table 5.3b.

Interestingly, for the local clusterings, each metric coheres with the final top-ranked approaches, i.e., all of the hyperparameter combinations are ranked very high for each of the chosen metrics. We believe this is due to the structure of the gold clustering, i.e., many clusters in total and most are relatively small, i.e., only a few elements are contained (cf. Figure 5.5 a & c). All of the top local clusterings are using the Markov Clustering algorithm (**mcl**) with the 'keep top 10 edges per node' pruning strategy (**top**$_{10}$). Also, the RoBERTa model seems to perform best, where only the representation of the feature embeddings differ slightly, i.e., using the **all** attention mask with or without building pseudo sentences, using the average (**avg**) or only the **first** wordpiece embedding, and using any combination of the final layers ([0,1,2]).

The top 20 global clusterings using only the top 20 local clusterings show no clearly distinctive or best clustering algorithm or pruning strategy (cf. Table 5.3b). Also, the final global clusterings range from rank 144 to 256, and the metrics do not necessarily correlate and mix within the top 20, e.g., while the AMI metric ranks the best clustering relatively high (336), the PUF_1 metric ranks it rather low (1, 160), and when PUF_1 ranks a clustering high (9), AMI ranks it low (1, 245). This shows again that there is no definitive 'best' clustering evaluation metric to trust.

5.4.6 Examples

We present hand-picked examples for the best clustering in Table 5.2, which demonstrate interesting phenomena, e.g., we observe clusters based on similarities in various levels of the language system such as semantic (Figure 5.6) or stylistic (Figure 5.7).

Figure 5.6 shows the clusterings (local and global) and gold clustering for the example verb 'run'. We can see that the largest cluster (G1) maps mainly to the frame 'self motion', mixed with 'quitting a place' and 'fleeing', which suggest that these verbs can be used to evoke semantic relations that involve some movement of a person. G2 exclusively maps to sentences that express frames about 'liquids', e.g., 'fluidic motion', 'cause to be wet', etc. However, cluster G3 shows mixed semantic relations such as 'operating a business' and 'operating a system'. Since this is the smallest cluster, we believe this is rooted in the fact that those frames do not come with enough sentences in the training data.

Figure 5.7 shows clusterings for the verb 'grunt'. Instead of showing the frame name, we show parts of the frame structure, i.e., its first argument type. Note that the entire structure of a frame is too fine-grained to be visualized as class labels. In the example, we can observe two global clusters. The second global cluster 'G2' is cleaner than 'G1' and shows nicely that the clustered sentences evoke 'Communication Noise' frames with 'Message' as the first argument. This suggests sentences being in passive voice, e.g., a sentence from this cluster is '"Huh!" he grunted disbelievingly.', i.e., the 'Message' beeing uttered by a speaker. In the larger cluster 'G1', we observe mixed sentences, but visually more sentences with an active voice, e.g., a sentence from this cluster is 'She grunted a reply and offered coffee.', i.e., a 'Speaker' uttering a message. This example shows nicely that our method also treats stylistic phenomena as semantic information. We believe this is due to the CWE representation.

5.5 Conclusion

We presented a resource-friendly clustering approach using contextualized embeddings. Since the list of contextual embeddings for all words of a corpus becomes too large to handle on a single computer or GPU, we used and adapted the local-global clustering as presented in (Ustalov et al., 2019).

The approach is re-implemented in PyTorch[5], has GPU support, and is available as an open-source package under a permissive license.[6] Due to the local-global clustering strategy, the approach is memory efficient. Samples are first collected for each verb type, then clustered in the local step – which can be done in parallel – and the global step then groups the local clusters and the elements within. Samples can be traced to their original occurrence in the corpus, i.e., the provenance of the input samples is provided.[7]. While the local clustering step addresses splitting of polysems, i.e., the instantiations of a particular word are grouped into different categories, which can roughly be resolved to different senses, the global clustering

5. https://pytorch.org/
6. https://github.com/remstef/loglo, Apache License V2.
7. A demo is available at http://ltdemos.informatik.uni-hamburg.de/unframes

step addresses the grouping of synonyms, i.e., the local groups (local senses) are joined with samples across different verbs.

We experimented with various pre-trained transformer models and their contextualized vector representations. Different masking strategies within the attention mechanism to include linguistic knowledge have been tested, and we conclude that the attention mechanism does not gain from explicit masked linguistic structures since the top clusterings utilize the **`all`** mask, where every token as access to every other token, which is the default strategy. In our unsupervised frame induction experiments, we evaluate clusterings produced by different parameterizations, compare them to FrameNet as gold clustering using several clustering evaluation metrics, and provide a set of hyperparameters that perform reasonably well. While we focused on verbs in this chapter, we note that the procedure itself is independent of the particular part of speech.

In the future, we plan to align the clusters with classes of a downstream task, e.g., with annotated relation classes, by exploiting the attention mechanism and using probabilities for words given a cluster as anchors. This would open new possibilities on traceability and interpretability of neural network models. Another line of research opens the possibility of semisupervised clustering, i.e., by adapting the label propagation procedure of Chinese Whispers to only update labels of unlabelled nodes, i.e., where class labels are not provided. This way, we could enrich a given annotated dataset with new examples and new classes.

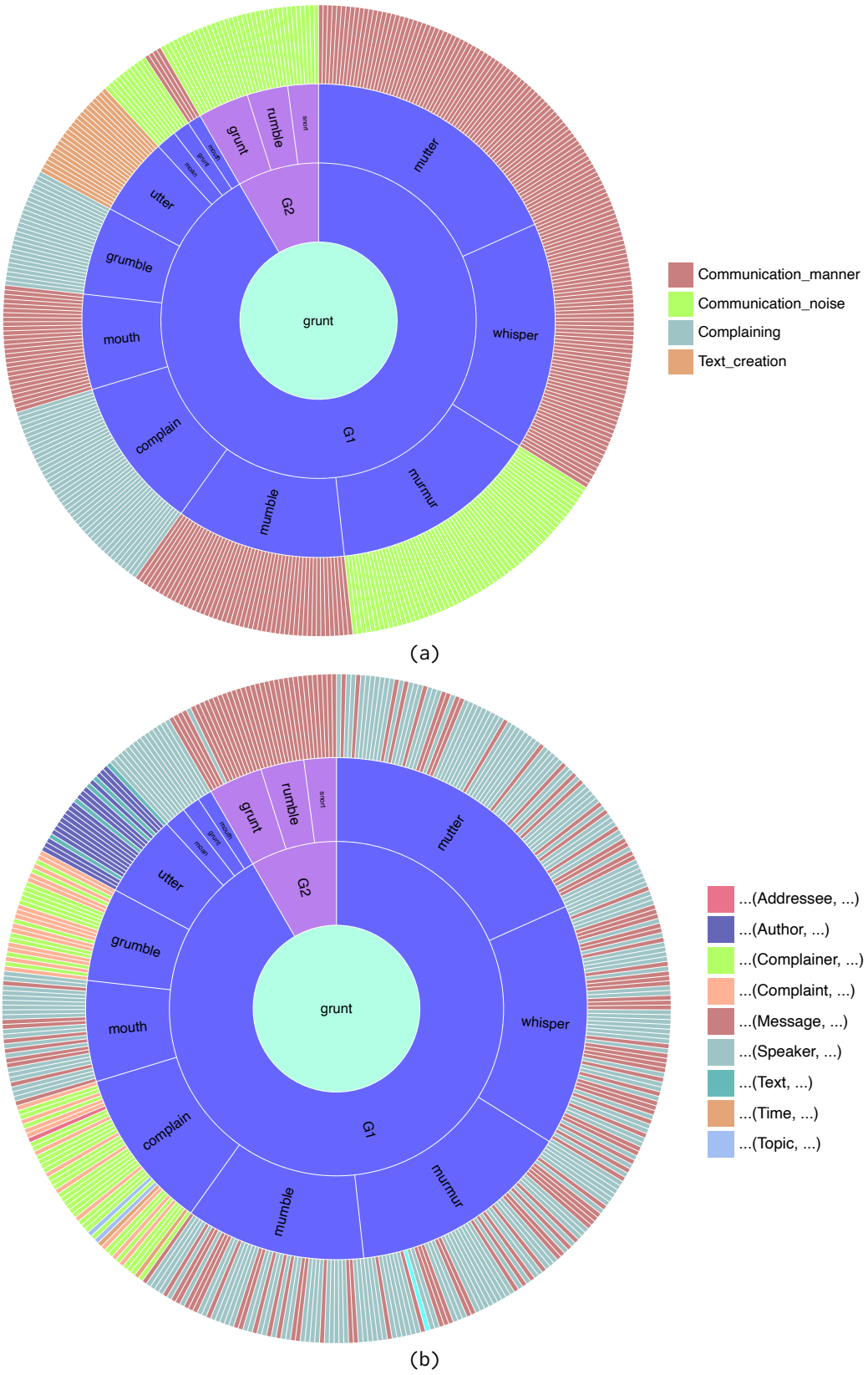


Figure 5.7: Final clusters where the word 'grunt' is involved and their labels as defined by the FrameNet. The outermost circle represents the sentences in which the respective word of the local clusters (second outermost circle) occur. For illustrative purposes, the legend covers the name (a) or the first argument type (b) of the frame, which is evoked by the verb in the sentence (only outermost colors).

6 Conclusion

The work in this dissertation is dedicated to two themes: *a*) the expansion of domain-dependent corpora, and *b*) the analysis of contextualized word embeddings for semantic structuring tasks. We highlighted the importance of domain dependent data for domain tasks, i.e., NLP tasks performed within a particular language domain. We note that context matters, both on the level of domain data, as well as more fine-grained on the vector representation level. In this chapter, we summarize the findings of this dissertation and highlight its contributions. Afterward, we conclude this dissertation with a summary and final remarks, including an outlook into possible future work.

6.1 Summary

Chapter 2 covered the expansion of domain corpora. We proposed a novel but simple approach to focused web-crawling based on the assumption that web links of a relevant webpage mainly point to other relevant web pages. Given a small initial domain defining corpus, the relevance of a URL is defined by the compatibility of its enclosed webpage to belong to that domain. The domain is modeled as a statistical language model – more specifically, we employed a Kneser-Ney model (Kneser and Ney, 1995) – which is built only from the given, initial corpus. The relevance of a web document is then measured in terms of perplexity with respect to the in-domain language model. The size of the corpus can be as small as a single webpage, e.g., among our experiments, we tested the approach using a single *featured Wikipedia article*.[1] Due to the limited corpus size, we argue that currently, more popular neural language models, such as transformer-based models, are not suited for this task because they require enormous amounts of data to perform well.

1. A featured article in Wikipedia is considered to be of high quality, suitable as a guide for writing and editing other articles. The textual content is usually larger than a random Wikipedia article.

We provide our focused web crawling system called TopiCrawler[2] as an open-source extension to the popular general web crawling framework Heritrix[3] (Mohr et al., 2004), which is used by the Internet Archive, a non-profit library of internet content, such as archival snapshots of websites and media content. Because of the dynamic nature of the web, evaluating web crawling methods, in general, is a non-trivial task. We thus evaluated our approach *a*) intrinsically, by comparing the perplexity scores of collected corpora with a test split of the source corpus, and *b*) extrinsically, by the downstream task application of Taxonomy Induction. Our findings show that in-domain corpora help for domain-dependent downstream tasks and that only a tiny fraction of in-domain data is needed to increase the entire in-domain data amount quickly and, by this, increase the performance of downstream tasks. We showed that it is possible to automatically extend corpora with focused crawling techniques that make use of language models. It is beneficial not only for constructing more accurate language models of the target domain but also for increasing performance for unsupervised tasks such as taxonomy induction for example, which require large (in-domain) corpora. In times where many NLP systems rely on large background corpora for pre-training or fine-tuning, computing word embeddings, or modeling language in general, focused crawling is a viable and straightforward way to grow existing domain-dependent background text collections.

Further, since a word can be interpreted and mapped to many senses, a single representation in the vector space is unreasonable. Chapter 3 covered modeling senses in the vector space instead of words. More specifically, we explained how different word senses can be modeled by distinct, static sense vectors instead of a single static word vector. We combined symbolic clustering methods with static word embeddings, tested clustering approaches in the word vector space directly, and analyzed the effectiveness of contextualized word embeddings for implicit sense modeling and explicit word sense disambiguation. Our results confirmed that sense embeddings do help to resolve word similarities, and we presented consistent improvements over all tested languages, embedding models, sense inventories, and datasets. Furthermore, we found that WSD can be surprisingly effective using CWEs from pre-trained models only. Our experiments revealed that contextual word embeddings (CWEs) are generally able to capture senses, i.e., words, when used in a different sense, are placed in different regions of the vector space.

Based on those findings, we used contextualized word embeddings to index and retrieve semantic relations in Chapter 4. We analyzed different explicit embedding extraction and aggregation strategies to include general linguistic knowledge and showed that more specific and task-dependent representations, such as the path in the dependency graph, help for retrieving similar sentences expressing the same semantic relation, which again helps for simple nearest neighbor classification in few-shot learning scenarios. Based on the success of this strategy, we continued this line of research and showed in Chapter 5 the unsupervised extraction of semantic relations. Because of computational benefits and the success in previous work, we used a local-global clustering approach to cluster occurrences of verbs into so-called relation clusters. With the local-global paradigm, we approached

2. https://tudarmstadt-lt.github.io/topicrawler/
3. http://crawler.archive.org

the ambiguity of verbs by explicitly modeling polysemy (local step) and synonymy (global step). We compare our fully unsupervised approach to the verb fraction of FrameNet, which we define as a relation repository and the gold clustering. Among our many experiments, we showed that it is possible to group instantiations of verbs, i.e., verbs occurring in a sentence, which evoke similar frames, in our case: relations. We found that similarities can be found on different levels of the language system; namely, we found clusters with semantically similar sentences, which was our initial hypothesis, but we also found sentences expressing similar behavior in terms of style. This comes from the representation provided by CWEs. Here, we also found that internal infusion of linguistic knowledge by adapting the attention mechanism of transformer-based models, which differs from our external knowledge infusion approach in Chapter 4, does not necessarily produce better clusterings, quantitatively and qualitatively.

6.2 Outlook (Future Work)

The central theme in this dissertation is concerned with the unsupervised extraction of information from text, tightly following Biemann's (2012) *structure discovery* (SD) paradigm. Supervised approaches are desirable for completely defined tasks, i.e., a clear target can be identified, data and annotations are not an issue, and the definition does not change over time. Because of this characterization, supervised approaches are usually more precise. But language evolves, changes, and more complex tasks often change over time, or annotations and data are just too expensive to come by. Here, unsupervised approaches can support the creative process of a basis that can be used and refined for supervised approaches.

Depending on the task, web crawling, in general, might appear obsolete due to the vast availability of datasets, e.g., the Common Crawl (CC)[4]. Even domain-dependent tasks can be addressed by filtering the vast general datasets for the domain of interest. However, there is often a lot of detail missing on the premises of the dataset creation, e.g., filtering certain websites, parsing details, etc. We argue that our approach for focused web-crawling can be used to gain the most recent information directly from the web, e.g., as needed by applications that report and rely on daily information such as the *network of the day*[5] (NOD; Benikova et al., 2014) or Storyfinder[6] (**Remus** et al., 2017). Filtering and other data-altering operations are fully transparent, and the collected data is up-to-date. Imagine domain-dependent applications handling daily data like NOD, an incremental learning paradigm, which updates its background language model for focused crawling on the basis of, e.g., yesterday's data, which might be a beneficial application. More extensions to this might even employ the *human in the loop* (HITL) paradigm, such as in Storyfinder.

Also, combining supervised and unsupervised approaches for a task might be a desirable extension. e.g., extending Chapter 5, we are exploring techniques for

4. https://commoncrawl.org/
5. http://tagesnetzwerk.de/
https://www.inf.uni-hamburg.de/en/inst/ab/lt/resources/demos/network-of-the-day.html
6. https://uhh-lt.github.io/storyfinder/

semi-supervised clustering of frames to enrich a given collection of sentences with frames with more sentences and potentially new, previously unknown frames.

Also, our sense induction technique might appear obsolete with the invention of CWEs, but we argue that inducing senses on a CWE level might also help for downstream tasks, and equally interesting is the analysis of language in general, i.e., independent of a downstream task, analyzing senses periodically using CWEs might reveal new insights on the use and generation of language.

Further, we analyzed CWEs and showed their superior performance for downstream tasks, but when computing similarities, they often indicate unclear or unexpected similarities, e.g., on a stylistic level than on a semantic level. *Constraint-based pre-training* of neural language models by adding more or other contexts, which might even be provided manually, could yield further insights and application domains, e.g., retrofitting text to a specific style like poems. Another line of active research is the interpretability of language models and neural language models in general, i.e., we advocate for more traceability and provenance of data samples and their influence in algorithmic decisions. This might involve explainability within a certain context, e.g., given context of some form, what are the reasoning steps of a model trained on generic data, 'does the context fit?'.

References

Eneko Agirre, Olatz Ansa, Eduard H Hovy, and David Martínez. 2000. Enriching very large ontologies using the WWW. In *ECAI Workshop on Ontology Learning,* 28–33. Berlin, Germany. (Cited on page 31).

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In *Proceedings of the 27th International Conference on Computational Linguistics,* 1638–1649. Santa Fe, NM, USA. (Cited on pages 15, 61, 64 sq., 75).

James Allen. 1995. Natural Language Understanding. Second. Redwood City: The Benjamin/Cummings Publishing Company Inc. (Cited on page 1).

Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. 1999. Modern Information Retrieval. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. (Cited on page 45).

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations,* 1–15. (Cited on pages 89, 98).

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics,* 86–90. Montréal, QC, Canada. (Cited on pages 90, 96).

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the Blanks: Distributional Similarity for Relation Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics,* 2895–2905. Florence, Italy. (Cited on page 77).

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (ACL),* 26–33. Toulouse, France. (Cited on page 19).

Marco Baroni and Silvia Bernardini. 2004. BootCaT: Bootstrapping Corpora and Terms from the Web. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC),* 1313–1316. Lisbon, Portugal. (Cited on pages 20, 34).

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *LRE* 43 (3): 209–226. (Cited on pages 19 sqq., 24).

Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov. 2016. Breaking sticks and ambiguities with adaptive skip-gram. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS),* 130–138. Cadiz, Spain. (Cited on pages 46 sq., 54, 64).

Siamak Barzegar, Brian Davis, Manel Zarrouk, Siegfried Handschuh, and Andre Freitas. 2018. SemR-11: A Multi-Lingual Gold-Standard for Semantic Similarity and Relatedness for Eleven Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).* Miyazaki, Japan: European Language Resources Association (ELRA), May. (Cited on page 57).

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *JMLR* 3:1137–1155. (Cited on page 23).

Darina Benikova, Uli Fahrer, Alexander Gabriel, Manuel Kaufmann, Seid Muhie Yimam, Tatiana von Landesberger, and Chris Biemann. 2014. Network of the day: Aggregating and visualizing entity networks from online sources. In *Proceedings of the Natural Language Processing for Computer-Mediated Communication / Social Media (NLP 4 CMC) workshop,* 48–52. Hildesheim, Germany. (Cited on page 109).

Douglas Biber. 1995. Dimensions of Register Variation: A Cross-Linguistic Comparison. Cambridge: Cambridge University Press. (Cited on page 21).

Chris Biemann. 2005. Ontology Learning from Text: A Survey of Methods. *LDV Forum* 20 (2): 75–93. (Cited on page 31).

———. 2006. Chinese Whispers - An Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proceedings of TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing,* 73–80. New York City, NY, USA. (Cited on pages 48, 59, 88, 94 sq.).

———. 2007. Unsupervised and Knowledge-free Natural Language Processing in the Structure Discovery Paradigm. Ph.D., Universität Leipzig. (Cited on page 95).

———. 2012. Structure Discovery in Natural Language. Berlin & Heidelberg, Germany: Springer Berlin Heidelberg. (Cited on pages 3, 95, 97, 109).

Chris Biemann, Felix Bildhauer, Stefan Evert, Dirk Goldhahn, Uwe Quasthoff, Roland Schäfer, Johannes Simon, Swiezinski Swiezinski, and Torsten Zesch. 2013. Scalable Construction of High-Quality Web Corpora. *Journal for Language Technology and Computational Linguistics (JLCL)* 27 (2). (Cited on pages 20 sq.).

Chris Biemann, Gerhard Heyer, and Uwe Quasthoff. 2022. Wissensrohstoff Text. Wiesbaden: Springer. (Cited on page 14).

Chris Biemann, Uwe Quasthoff, Gerhard Heyer, and Florian Holz. 2008. ASV Toolbox: a Modular Collection of Language Exploration Tools. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC),* 1760–1767. Marrakech, Morocco. (Cited on page 29).

Chris Biemann and Martin Riedl. 2013. Text: Now in 2D! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling* 1 (1): 55–95. (Cited on page 48).

Steven Bird, Ewan Klein, and Edward Loper. 2009. Natural language processing with Python: analyzing text with the natural language toolkit. Sebastopol, CA, USA: O'Reilly Media, Inc. (Cited on page 96).

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3:993–1022. (Cited on page 14).

Leonard Bloomfield. 1933. Language. Holt, Rinehardt / Winston, New York, NY, USA. (Cited on page 7).

Avrim Blum and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory,* 92–100. Madison, Wisconsin, USA. (Cited on pages 21, 24).

Georgeta Bordea, Paul Buitelaar, Stefano Faralli, and Roberto Navigli. 2015. SemEval-2015 Task 17: Taxonomy Extraction Evaluation (TExEval). In *Proceedings of the 9th International Workshop on Semantic Evaluation,* 902–910. Denver, CO, USA. (Cited on pages 32, 34, 38).

Georgeta Bordea, Els Lefever, and Paul Buitelaar. 2016. SemEval-2016 Task 13: Taxonomy Extraction Evaluation (TExEval-2). In *Proceedings of the 10th International Workshop on Semantic Evaluation,* 1081–1091. San Diego, CA, USA. (Cited on page 32).

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram Version 1. Philadelphia: Linguistic Data Consortium. (Cited on page 19).

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165,* (cited on page 19).

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal Distributional Semantics. *Journal of Artificial Intelligence Research* (USA) 49 (1): 1–47. (Cited on pages 46, 50).

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram Counts and Language Models from the Common Crawl. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC),* 3579–3584. Reykjavik, Iceland. (Cited on page 20).

Razvan Bunescu and Raymond Mooney. 2005. A Shortest Path Dependency Kernel for Relation Extraction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing,* 724–731. Vancouver, BC, Canada. (Cited on page 80).

Curt Burgess. 1998. From simple associations to the building blocks of language: Modeling meaning in memory with the HAL model. *Behavior Research Methods, Instruments, & Computers* 30 (2): 188–198. (Cited on page 51).

Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. 2009. The ClueWeb09 Dataset. http://lemurproject.org/about.php. (Cited on pages 20, 24).

Jose Camacho-Collados and Mohammad Taher Pilehvar. 2018. From Word to Sense Embeddings: A Survey on Vector Representations of Meaning. *Journal of Artificial Intelligence Research* 63 (1): 743–788. (Cited on page 65).

Lyle Campbell. 2017. The History of Linguistics. Chap. 6 in *The Handbook of Linguistics,* 95–117. John Wiley & Sons, Ltd. (Cited on page 3).

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI).* Atlanta, GA, USA. (Cited on page 19).

Soumen Chakrabarti, Martin van den Berg, and Byron Dom. 1999. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks* 31 (11–16): 1623–1640. (Cited on pages 20 sq., 24, 26).

Stanley F Chen and Joshua Goodman. 1998. An Empirical Study of Smoothing Techniques for Language Modeling. Technical report TR-10-98. Harvard University. (Cited on pages 22 sq.).

Noam Chomsky. 1957. Syntactic Structures. *Language* 33 (3): 1–375. (Cited on page 6).

———. 1965. Aspects of the Theory of Syntax. MIT press, Cambridge, MA, USA. (Cited on page 6).

Philipp Cimiano, Andreas Hotho, and Steffen Staab. 2005. Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis. *Journal of Artificial Intelligence Research* 24 (1): 305–339. (Cited on page 32).

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What Does BERT Look at? An Analysis of BERT's Attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP,* 276–286. Florence, Italy. (Cited on page 77).

Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13 (1): 21–27. (Cited on page 66).

Michael Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation,* 376–380. Baltimore, MD, USA. (Cited on page 46).

Ferdinand de Saussure. 1983; orig. 1916. Course in General Linguistics. Edited by Charles Bally, Albert Riedlinger, and Albert Sechehaye. Duckworth & Co, London, UK. (Cited on pages 5, 63).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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,* 4171–4186. Minneapolis, MN, USA. (Cited on pages 11, 15, 19, 21, 23, 61, 65 sq., 75, 77, 91, 98).

Beate Dorow and Dominic Widdows. 2003. Discovering Corpus-Specific Word Senses. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics.* Budapest, Hungary. (Cited on pages 47, 49, 88).

Philip Edmonds and Scott Cotton. 2001. SENSEVAL-2: Overview. In *Proceedings of SENSEVAL-2: Second International Workshop on Evaluating Word Sense Disambiguation Systems,* 1–5. Toulouse, France. (Cited on page 63).

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14 (2): 179–211. (Cited on page 77).

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining,* 226–231. Portland, OR, USA. (Cited on page 94).

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. 2008. Open Information Extraction from the Web. *Communications of the ACM* (New York, NY, USA) 51 (12): 68–74. (Cited on page 19).

Stefano Faralli, Giovanni Stilo, and Paola Velardi. 2015. Large Scale Homophily Analysis in Twitter Using a Twixonomy. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015,* 2334–2340. Buenos Aires, Argentina. (Cited on page 34).

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting Word Vectors to Semantic Lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 1606–1615. Denver, CO, USA. (Cited on page 47).

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems With Evaluation of Word Embeddings Using Word Similarity Tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP,* 30–35. Berlin, Germany. (Cited on page 52).

Christiane Fellbaum. 1998. WordNet: An Electronic Lexical Database. The MIT Press. (Cited on pages 21, 46, 61, 65, 67).

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web,* 406–414. Hong Kong, Hong Kong. (Cited on pages 46, 50, 57).

John R Firth. 1957. A Synopsis of Linguistic Theory, 1930-1955. *Studies in Linguistic Analysis,* (cited on page 7).

Gottlob Frege. 1892. Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik* (Berlin, Germany) 100 (1): 25–50. (Cited on page 8).

Brendan J Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *Science* 315 (5814): 972–976. (Cited on page 94).

Kata Gábor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Haïfa Zargayouna, and Thierry Charnois. 2018. SemEval-2018 Task 7: Semantic Relation Extraction and Classification in Scientific Papers. In *Proceedings of The 12th International Workshop on Semantic Evaluation,* 679–688. New Orleans, LA, USA. (Cited on page 81).

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL-HLT,* 758–764. Atlanta, GA, USA, June. (Cited on page 51).

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* 2173–2182. Austin, TX, USA. (Cited on pages 46, 50).

Eleanor J Gibson. 1971. Perceptual learning and the theory of word perception. *Cognitive Psychology* 2 (4): 351–368. (Cited on page 2).

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. SemEval-2007 Task 04: Classification of Semantic Relations between Nominals. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007),* 13–18. Prague, Czech Republic. (Cited on page 81).

Dirk Goldhahn, **Steffen Remus**, Uwe Quasthoff, and Chris Biemann. 2014. Top-Level Domain Crawling for Producing Comprehensive Monolingual Corpora from the Web. In *Proceedings of the LREC-14 workshop on Challenges in the Management of Large Corpora (CMLC-2),* 10–14. Reykjavik, Iceland. (Cited on pages 20, 28).

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. Cambridge, MA, USA: MIT Press. (Cited on page 15).

Joshua T Goodman. 2001. A Bit of Progress in Language Modeling. Technical report MSR-TR-2001-72. Microsoft. (Cited on page 22).

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning Word Vectors for 157 Languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018).* (Cited on page 57).

Richard L Gregory. 1980. Perceptions as Hypotheses. *Philosophical Transactions of the Royal Society of London,* B, Biological Sciences, 290 (1038): 181–197. (Cited on page 2).

Zenzi M Griffin and Victor S Ferreira. 2006. Properties of Spoken Language Production. In *Handbook of Psycholinguistics (Second Edition),* Second Edition, edited by Matthew J. Traxler and Morton A. Gernsbacher, 21–59. London: Academic Press. (Cited on page 9).

Fritz Günther, Carolin Dudschig, and Barbara Kaup. 2015. LSAfun - An R package for computations based on Latent Semantic Analysis. *Behavior Research Methods* 47 (4): 930–944. (Cited on page 51).

Alon Halevy, Peter Norvig, and Fernando Pereira. 2009. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems* 24 (2): 8–12. (Cited on page 19).

Zellig S Harris. 1951. Methods in Structural Linguistics. University of Chicago Press, Chicago, IL, USA. (Cited on pages 7, 63).

———. 1954. Distributional Structure. *Word* 10 (23): 146–162. (Cited on pages 7, 13, 45, 61, 76).

Marti A Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 15th Conference on Computational Linguistics (Coling),* 539–545. Nantes, France. (Cited on pages 31 sqq.).

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation,* 33–38. Uppsala, Sweden. (Cited on page 81).

Felix Hill, Kyunghyun Cho, Sébastien Jean, Coline Devin, and Yoshua Bengio. 2014. Embedding Word Similarity with Neural Machine Translation. *CoRR* abs/1412.6448. (Cited on pages 46, 50).

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. *Computational Linguistics* 41 (4): 665–695. (Cited on page 57).

Thomas Hofmann. 1999. Probabilistic Latent Semantic Analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence,* 289–296. Stockholm, Sweden. (Cited on page 14).

Matthew Honnibal and Mark Johnson. 2015. An Improved Non-monotonic Transition System for Dependency Parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* 1373–1378. Lisbon, Portugal. (Cited on page 66).

David Hope and Bill Keller. 2013. MaxMax: A Graph-based Soft Clustering Algorithm Applied to Word Sense Induction. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I,* 368–381. Samos, Greece: Springer-Verlag. (Cited on pages 88, 94).

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R. Bowman. 2019. Do Attention Heads in BERT Track Syntactic Dependencies? In *Natural Language, Dialog and Speech (NDS) Symposium,* 1–7. New York, NY, USA. (Cited on pages 76 sq.).

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What Does BERT Learn about the Structure of Language? In *57th Annual Meeting of the Association for Computational Linguistics,* 3651–3657. Florence, Italy. (Cited on page 70).

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics* (Cambridge, MA) 8:64–77. (Cited on page 98).

Mikael Kågebäck and Hans Salomonsson. 2016. Word Sense Disambiguation using a Bidirectional LSTM. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex-V),* 51–56. Osaka, Japan. (Cited on pages 65, 69).

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing Word Embeddings for Similarity or Relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* 2044–2048. Lisbon, Portugal. (Cited on page 47).

Adam Kilgarriff. 2001. English Lexical Sample Task Description. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems,* 17–20. Toulouse, France. (Cited on pages 63, 67).

———. 2007. Googleology is Bad Science. *Computational Linguististics (CL)* 33 (1): 147–151. (Cited on page 20).

Jon M Kleinberg. 1999. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM (JACM)* (New York, NY, USA) 46 (5): 604–632. (Cited on page 37).

Reinhard Kneser and Hermann Ney. 1995. Improved Backing-off for M-Gram Language Modeling. In *ICASSP,* 181–184. (Cited on pages 22, 107).

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics,* 177–180. Prague, Czech Republic. (Cited on page 22).

Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on Web search and data mining (WSDM),* 441–450. New York, NY, USA. (Cited on page 23).

Bernd Kortmann. 2005. English Linguistics: Essentials. Cornelsen Verlag, Berlin, Germany. (Cited on pages 5 sq.).

Zornitsa Kozareva and Eduard Hovy. 2010. Learning Arguments and Supertypes of Semantic Relations Using Recursive Patterns. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL),* 1482–1491. Uppsala, Sweden. (Cited on page 32).

Roland Kuhn and Renato de Mori. 1990. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 12 (6): 570–583. (Cited on page 22).

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to Plato's problem: the Latent Semantic Analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review* 104:211–240. (Cited on page 14).

Omer Levy, **Steffen Remus**, Chris Biemann, and Ido Dagan. 2015. Do Supervised Distributional Methods Really Learn Lexical Inference Relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 970–976. Denver, CO, USA. (Cited on page 63).

Dekang Lin and Patrick Pantel. 2001. DIRT - Discovery of Inference Rules from Text. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining (KDD),* 323–328. San Francisco, CA, USA. (Cited on page 80).

Nelson F Liu, Matt Gardner, Yonatan Belinkov, Matthew E Peters, and Noah A Smith. 2019. Linguistic Knowledge and Transferability of Contextual Representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 1073–1094. Minneapolis, MN, USA. (Cited on page 91).

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692. arXiv: 1907.11692Liu.2019. (Cited on pages 79, 98).

Varvara Logacheva, Denis Teslenko, Artem Shelmanov, **Steffen Remus**, Dmitry Ustalov, Andrey Kutuzov, Ekaterina Artemova, Chris Biemann, and Alexander Panchenko. 2020. Word sense disambiguation for 158 languages using word embeddings only. In *Proceedings of The 12th Language Resources and Evaluation Conference,* 5943–5952. Marseille, France. (Cited on pages 57 sq., 64).

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9 (11): 2579–2605. (Cited on pages 55, 70).

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Introduction to Information Retrieval. Cambridge University Press. (Cited on page 25).

Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 1999. Building domain-specific search engines with machine learning techniques. In *Proceedings of the AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace,* 662–667. Orlando, FL, USA. (Cited on pages 20 sq., 24).

Olena Medelyan, Stefan Schulz, Jan Paetzold, Michael Poprat, and Kornél Markó. 2006. Language Specific and Topic Focused Web Crawling. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC),* 865–868. Genoa, Italy. (Cited on pages 20 sq.).

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning Generic Context Embedding with Bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning,* 51–61. Berlin, Germany. (Cited on page 65).

Filippo Menczer, Gautam Pant, and Padmini Srinivasan. 2004. Topical Web Crawlers: Evaluating Adaptive Algorithms. *ACM Transactions Internet Technology (TOIT)* (New York, NY, USA) 4 (4): 378–419. (Cited on pages 21, 26).

Robert Meusel, Peter Mika, and Roi Blanco. 2014. Focused Crawling for Structured Data. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management,* 1039–1048. CIKM '14. Shanghai, China. (Cited on page 26).

Rada Mihalcea, Timothy Chklovski, and Adam Kilgarriff. 2004. The Senseval-3 English lexical sample task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text,* 25–28. Barcelona, Spain. (Cited on pages 63, 67).

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the 1st International Conference on Learning Representations.* Scottsdale, AZ, USA. (Cited on pages 14 sq., 47, 50, 76).

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018).* (Cited on page 57).

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 746–751. Atlanta, GA, USA: Association for Computational Linguistics. (Cited on pages 15, 45, 58).

Stanley Milgram. 1967. The Small World Problem. *Psychology Today* 67 (1): 61–67. (Cited on page 93).

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes (LCP)* 6 (1): 1–28. (Cited on page 57).

George A Miller, Eugene Galanter, and Karl H Pribram. 1960. Plans and the Structure of Behaviour. In *Systems Research for Behavioral Science,* 1st Edition, missing 14. New York, NY, USA: Henry Holt / Co. (Cited on page 9).

George A Miller, Claudia Leacock, Randee Tengi, and Ross T Bunker. 1993. A Semantic Concordance. In *Proceedings of the Workshop on Human Language Technology,* 303–308. HLT '93. Princeton, NJ, USA. (Cited on pages 65, 67).

Gordon Mohr, Michele Kimpton, Micheal Stack, and Igor Ranitovic. 2004. Introduction to heritrix, an archival quality web crawler. In *Proceedings of the 4th International Web Archiving Workshop IWAW'04,* 1–15. Bath, UK. (Cited on pages 24, 108).

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting Word Vectors to Linguistic Constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 142–148. San Diego, CA, USA. (Cited on page 51).

Jinseok Nam, Christian Kirschner, Zheng Ma, Nicolai Erbs, Susanne Neumann, Daniela Oelke, **Steffen Remus**, Chris Biemann, Judith Eckle-Kohler, Johannes Fürnkranz, Iryna Gurevych, Marc Rittberger, and Karsten Weihe. 2014. Knowledge Discovery in Scientific Literature. In *Proceedings of the 12th Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2014),* 66–76. Hildesheim, Germany. (Cited on page 27).

Vivi Nastase, Preslav Nakov, Diarmuid Ó Séaghdha, and Stan Szpakowicz. 2013. Semantic Relations Between Nominals. In *Synthesis Lectures on Human Language Technologies,* vol. 6. 1. Morgan & Caypool Publishers. (Cited on page 87).

Roberto Navigli. 2009. Word Sense Disambiguation: A Survey. *ACM Computing Surveys (CSUR)* (New York, NY, USA) 41, no. 2 (February): 10:1–10:69. (Cited on page 46).

Roberto Navigli, Kenneth C. Litkowski, and Orin Hargraves. 2007. SemEval-2007 Task 07: Coarse-Grained English All-Words Task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007),* 30–35. Prague, Czech Republic. (Cited on page 67).

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* 1059–1069. Doha, Qatar. (Cited on pages 47, 64).

Nils John Nilsson. 2010. The Quest for Artificial Intelligence: A History of Ideas and Achievements. 1st edition. New York, NY, USA: Cambridge University Press. (Cited on page 1).

Michael P Oakes. 2005. Using Hearst's Rules for the Automatic Acquisition of Hyponyms for Mining a Pharmaceutical Corpus. In *RANLP Text Mining Workshop'05,* 63–67. Borovets, Bulgaria. (Cited on page 31).

Charles K Ogden and Ivor A Richards. 1923. The Meaning of Meaning: A Study of the Influence of Thought and of the Science of Symbolism. Harcourt, Brace & World, Inc. (Cited on pages 8, 10).

OpenAI. 2023. GPT-4 Technical Report. arXiv: 2303.08774 [cs.CL]. (Cited on page 19).

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66. Stanford InfoLab. (Cited on page 93).

Alexander Panchenko, Stefano Faralli, Eugen Ruppert, **Steffen Remus**, Hubert Naets, Cedrick Fairon, Simone P Ponzetto, and Chris Biemann. 2016. TAXI at SemEval-2016 Task 13: A Taxonomy Induction Method based on Lexico-Syntactic Patterns, Substrings and Focused Crawling. In *Proceedings of the 10th International Workshop on Semantic Evaluation,* 1320–1327. San Diego, CA, USA. (Cited on pages 20, 30, 32 sqq.).

Alexander Panchenko, Fide Marten, Eugen Ruppert, Stefano Faralli, Dmitry Ustalov, Simone Paolo Ponzetto, and Chris Biemann. 2017. Unsupervised, Knowledge-Free, and Interpretable Word Sense Disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations,* 91–96. Copenhagen, Denmark. (Cited on page 64).

Alexander Panchenko, Olga Morozova, and Hubert Naets. 2012. A semantic similarity measure based on lexico-syntactic patterns. In *Proceedings of the 11th Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2012),* 174–178. Vienna, Austria. (Cited on page 33).

Alexander Panchenko, Eugen Ruppert, Stefano Faralli, Simone Paolo Ponzetto, and Chris Biemann. 2018. Building a Web-Scale Dependency-Parsed Corpus from CommonCrawl. In *Proceedings of the 11th International Conference on Language Resources and Evaluation.* Miyazaki, Japan. (Cited on page 20).

Alexander Panchenko, Eugen Ruppert, Stefano Faralli, Paolo Simone Ponzetto, and Chris Biemann. 2017. Unsupervised Does Not Mean Uninterpretable: The Case for Word Sense Induction and Disambiguation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers,* 86–98. Valencia, Spain. (Cited on page 54).

Gautam Pant, Kostas Tsioutsiouliklis, Judy Johnson, and C Lee Giles. 2004. Panorama: Extending Digital Libraries with Topical Crawlers. In *Joint Conference on Digital Libraries (JCDL),* 142–150. Tuscon, AZ, USA. (Cited on page 20).

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge Discovery and Data mining (KDD),* 613–619. Edmonton, AB, Canada. (Cited on page 47).

Marius Pasca. 2004. Acquisition of categorized named entities for web search. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management (CIKM),* 137–145. Washington, D.C., USA. (Cited on page 32).

Michel Pêcheux. 2022. Automatic Discourse Analysis. Leiden, Netherlands: Brill. (Cited on page 63).

Maria Pelevina, Nikolay Arefiev, Chris Biemann, and Alexander Panchenko. 2016. Making Sense of Word Embeddings. In *Proceedings of the 1st Workshop on Representation Learning for NLP,* 174–183. Berlin, Germany. (Cited on pages 47, 64).

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* 1532–1543. Doha, Qatar. (Cited on pages 14, 50, 65, 76).

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 2227–2237. New Orleans, LA, USA. (Cited on pages 16, 19, 61, 65 sq., 75, 77).

Steven Pinker. 1994. The Language Instinct: How the Mind Creates. Harper Collins, New York, NY, USA. (Cited on page 7).

Jan Pomikálek, Miloš Jakubíček, and Pavel Rychlý. 2012. Building a 70 billion word corpus of English from ClueWeb. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC),* 502–506. Istanbul, Turkey. (Cited on page 20).

Hoifung Poon and Pedro Domingos. 2010. Unsupervised ontology induction from text. In *Proceedings of ACL 2010,* 296–305. Uppsala, Sweden. (Cited on page 32).

Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. SemEval-2007 Task-17: English Lexical Sample, SRL and All Words. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007),* 87–92. Prague, Czech Republic. (Cited on page 67).

Jialun Qin, Yilu Zhou, and Michael Chau. 2004. Building Domain-Specific Web Collections for Scientific Digital Libraries: A Meta-Search Enhanced Focused Crawling Method. In *JCDL,* 135–141. (Cited on page 20).

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. OpenAI Blog, (cited on page 19).

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. OpenAI Blog, (cited on page 19).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21 (140): 1–67. (Cited on page 19).

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers,* 99–110. Valencia, Spain. (Cited on page 63).

Gábor Recski, Eszter Iklódi, Katalin Pajkossy, and Andras Kornai. 2016. Measuring Semantic Similarity of Words Using Concept Networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP,* 193–200. Berlin, Germany. (Cited on page 51).

Nils Reimers and Iryna Gurevych. 2019. 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),* 3982–3992. Hong Kong. (Cited on page 77).

Steffen Remus. 2012. Automatically Identifying Lexical Chains by Means of Statistical Methods – A Knowledge-Free Approach. MA, Technische Universität Darmstadt. (Cited on pages 99 sq.).

———. 2014. Unsupervised Relation Extraction of In-Domain Data from Focused Crawls. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics,* 11–20. Gothenburg, Sweden. (Cited on pages 20, 80).

Steffen Remus and Chris Biemann. 2013. Three Knowledge-Free Methods for Automatic Lexical Chain Extraction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 989–999. Atlanta, GA, USA. (Cited on page 99).

———. 2016. Domain-Specific Corpus Expansion with Focused Webcrawling. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016),* 23–28. Portorož, Slovenia. (Cited on page 20).

———. 2018. Retrofitting Word Representations for Unsupervised Sense Aware Word Similarities. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018),* 1035–1041. Miyazaki, Japan. (Cited on pages 46, 57, 64).

Steffen Remus, Gerold Hintz, Darina Benikova, Thomas Arnold, Judith Eckle-Kohler, Christian M Meyer, Margot Mieskes, and Chris Biemann. 2016. EmpiriST: AIPHES Robust Tokenization and POS-Tagging for Different Genres. In *Proceedings of the 10th Web as Corpus Workshop (WAC-X),* 106–114. Berlin, Germany. (Cited on pages 20, 23).

Steffen Remus, Manuel Kaufmann, Kathrin Ballweg, Tatiana von Landesberger, and Chris Biemann. 2017. Storyfinder: Personalized Knowledge Base Construction and Management by Browsing the Web. In *CIKM '17: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management,* 2519–2522. Singapore, Singapore. (Cited on page 109).

Steffen Remus, Gregor Wiedemann, Saba Anwar, Fynn Petersen-Frey, Seid Muhie Yimam, and Chris Biemann. 2022. More Like This: Semantic Retrieval with Linguistic Information. In *Proceedings of the 18th Conference on Natural Language Processing (KONVENS 2022),* 156–166. Potsdam, Germany. (Cited on pages 75, 84, 89).

Martin Riedl. 2016. Unsupervised Methods for Learning and Using Semantics of Natural Language. PhD diss., Technische Universität. (Cited on pages 15, 48, 52).

Alan Ritter, Stephen Soderland, and Oren Etzioni. 2009. What is this, anyway: Automatic hypernym discovery. In *Proceedings of the 2009 AAAI Spring Symposium on Learning by Reading and Learning to Read,* 88–93. Palo Alto, CA, USA. (Cited on page 31).

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics* (Cambridge, MA, USA) 8:842–866. (Cited on page 91).

Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending Word Embeddings to Embeddings for Synsets and Lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing.* 1793–1803. Beijing, China. (Cited on pages 46 sq., 54, 64).

Herbert Rubenstein and John B. Goodenough. 1965. Contextual Correlates of Synonymy. *Commun. ACM* (New York, NY, USA) 8, no. 10 (October): 627–633. (Cited on page 57).

Sebastian Ruder. 2019. Neural Transfer Learning for Natural Language Processing. PhD diss., National University of Ireland, Galway. (Cited on page 15).

Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer Learning in Natural Language Processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics,* 15–18. Minneapolis, MN, USA. (Cited on page 15).

Eugen Ruppert, Manuel Kaufmann, Martin Riedl, and Chris Biemann. 2015. JoBimViz: A Web-based Visualization for Graph-based Distributional Semantic Models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing,* 103–108. Beijing, China. (Cited on page 49).

Mejdl S Safran, Abdullah Althagafi, and Dunren Che. 2012. Improving Relevance Prediction for Focused Web Crawlers. In *Proceedings of the 13th International Conference on Computer and Information Science,* 161–166. (Cited on pages 21, 24, 26).

Roland Schäfer, Adrien Barbaresi, and Felix Bildhauer. 2014. Focused Web Corpus Crawling. In *Proceedings of the 9th Web as Corpus Workshop (WaC-9),* 9–15. Gothenburg, Sweden. (Cited on pages 20, 26).

Roland Schäfer and Felix Bildhauer. 2012. Building Large Corpora from the Web Using a New Efficient Tool Chain. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC),* 486–493. Istanbul, Turkey. (Cited on pages 20, 24).

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* 298–307. Lisbon, Portugal. (Cited on pages 52, 54).

Hinrich Schütze. 1992a. Dimensions of Meaning. In *Proceedings of the 1992 ACM/IEEE Conference on Supercomputing,* 787–796. Minneapolis, MN, USA. (Cited on page 14).

———. 1992b. Word Space. In *Advances in Neural Information Processing Systems,* 5:895–902. Denver, CO, USA. (Cited on page 14).

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric Pattern Based Word Embeddings for Improved Word Similarity Prediction. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning,* 258–267. Beijing, China. (Cited on pages 50 sq.).

John R Searle. 1975. A Taxonomy of Illocutionary Acts. In *Language, Mind and Knowledge,* edited by Keith Gunderson, 344–369. University of Minnesota Press. (Cited on page 9).

Claude E Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal* 27 (3): 379–423. (Cited on page 11).

Stuart C Shapiro. 1987. Processing, bottom-up and top-down. In *Encyclopedia of Artificial Intelligence,* edited by Stuart C Shapiro, 1229–1234. Reprinted in Second Edition, 1992. John Wiley & Sons, Ltd. (Cited on page 2).

Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (8): 888–905. (Cited on page 94).

Rion Snow, Dan Jurafsky, and Andrew Y Ng. 2006. Semantic taxonomy induction from heterogeneous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL),* 801–808. Sydney, Australia. (Cited on page 32).

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2004. Learning Syntactic Patterns for Automatic Hypernym Discovery. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS),* 1297–1304. Vancouver, BC, Canada. (Cited on page 31).

Benjamin Snyder and Martha Palmer. 2004. The English all-words task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text,* 41–43. Barcelona, Spain. (Cited on page 63).

John F Sowa. 2000. Knowledge representation logical, philosophical, and computational foundations. Pacific Grove, CA, USA: Brooks/Cole. (Cited on pages 2, 30).

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing,* 5027–5038. Brussels, Belgium. (Cited on page 76).

Kaveh Taghipour and Hwee Tou Ng. 2015. Semi-Supervised Word Sense Disambiguation Using Word Embeddings in General and Specific Domains. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 314–323. Denver, CO, USA. (Cited on page 65).

Ming Tan, Wenli Zhou, Lei Zheng, and Shaojun Wang. 2012. A Scalable Distributed Syntactic, Semantic, and Lexical Language Model. *Compuational Linugistics* 38 (3): 631–671. (Cited on page 23).

Jie Tang, Ho fung Leung, Qiong Luo, Dewei Chen, and Jibin Gong. 2009. Towards Ontology Learning from Folksonomies. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI),* 2089–2094. Pasadena, CA, USA. (Cited on page 32).

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics,* 4593–4601. Florence, Italy. (Cited on page 91).

Alan M Turing. 1948. Intelligent Machinery. Technical report 10. (Cited on page 2).

———. 1950. Computing Machinery and Intelligence. *MIND: A Quarterly Review of Pyschology and Philosophy* 59 (236): 433–460. (Cited on page 1).

Dmitry Ustalov, Alexander Panchenko, Chris Biemann, and Simone Paolo Ponzetto. 2019. Watset: Local-Global Graph Clustering with Applications in Sense and Frame Induction. *Computational Linguistics* 45 (3): 423–479. (Cited on pages 87 sq., 103).

Stjin M van Dongen. 2000. Graph clustering. Ph.D., University of Utrecht. (Cited on pages 48, 88, 93).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Proceedings of Advances in Neural Information Processing Systems,* 5998–6008. Long Beach, CA, USA. (Cited on pages 15, 66, 75, 89, 98).

Loïc Vial, Benjamin Lecouteux, and Didier Schwab. 2018a. Improving the Coverage and the Generalization Ability of Neural Word Sense Disambiguation through Hypernymy and Hyponymy Relationships. *CoRR* abs/1811.00960. (Cited on pages 65, 69).

———. 2018b. UFSAC: Unification of Sense Annotated Corpora and Tools. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).* Miyazaki, Japan. (Cited on pages 63, 67).

———. 2019. Sense Vocabulary Compression through the Semantic Knowledge of WordNet for Neural Word Sense Disambiguation. *CoRR* abs/1905.05677. arXiv: 1905.05677. (Cited on pages 65, 69 sq.).

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Jiangnan Xia, Liwei Peng, and Luo Si. 2019. StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding. In *Proceedings of the International Conference on Learning Representations (ICLR).* Addis Ababa, Ethiopia (online). (Cited on page 77).

Gregor Wiedemann, **Steffen Remus**, Avi Chawla, and Chris Biemann. 2019. Does BERT Make Any Sense? Interpretable Word Sense Disambiguation with Contextualized Embeddings. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019): Long Papers,* 161–170. Erlangen, Germany. (Cited on pages 61, 64, 66, 77).

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From Paraphrase Database to Compositional Paraphrase Model and Back. *Transactions of the Association for Computational Linguistics* 3:345–358. (Cited on pages 47, 51).

———. 2016. Charagram: Embedding Words and Sentences via Character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* 1504–1515. Austin, TX, USA, November. (Cited on page 51).

Ludwig Wittgenstein. 1953. Philosophical Investigations. Transl. from Philosophische Untersuchungen. Macmillan. (Cited on page 10).

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed Masking: Parameter-free Probing for Analyzing and Interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* 4166–4176. Online. (Cited on page 77).

Hui Yang and Jamie Callan. 2009. A metric-based framework for automatic taxonomy induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics (ACL) and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (IJCNLP),* 271–279. Suntec, Singapore. (Cited on pages 31 sq.).

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised Word Sense Disambiguation with Neural Models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,* 1374–1385. Osaka, Japan. (Cited on pages 65, 69).

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation,* 75–86. Minneapolis, MN, USA. (Cited on page 64).

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. *arXiv preprint arXiv:2303.18223,* (cited on page 16).

Index

"A dissertation is not a sprint, it is a marathon."

— Chris Biemann (Personal Communications)