Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Adaptive Approaches to Natural Language Processing in Annotation and Application

Dissertation

zur Erlangung des akademischen Grades
Doctor rerum naturalium
(Dr.rer.nat.)
an der Fakultät für Mathematik, Informatik und Naturwissenschaften
der Universität Hamburg

Eingereicht beim Fachbereich Informatik von

Seid Muhie Yimam
aus Dessie (Äthiopien)

Hamburg
2019

Gutachterinnen/Gutachter

1. Prof. Dr. Chris Biemann

2. Prof. Dr.-Ing. Wolfgang Menzel

Tag der Disputation: 03.07.2019

Dedicated to my father (ጋሽዬ) and to my mother (አለምዬ) for believing in me and helping me succeed!

Thesis-Related Publication List

Book Chapters

1. Chris Biemann, Kalina Bontcheva, Richard Eckart de Castilho, Iryna Gurevych, and **Seid Muhie Yimam**. Collaborative Web-Based Tools for Multi-layer Text Annotation. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, pages 229–256. Dordrecht, 2017. preprint: https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2017-biemannetal-hola-webbasedtools-preprint.pdf

Journal Publications

1. **Seid Muhie Yimam**, Chris Biemann, Ljiljana Majnarić, Šefket Šabanović, and Andreas Holzinger. An adaptive annotation approach for biomedical entity and relation recognition. *Brain Informatics*, 3(3):157–168, 2016a. Online: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4999566/

Conference Proceedings

1. Gregor Wiedemann, **Seid Muhie Yimam**, and Chris Biemann. A Multilingual Information Extraction Pipeline for Investigative Journalism. In *Proceedings of 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, Brussels, Belgium, 2018a. Online: http://aclweb.org/anthology/D18-2014

2. **Seid Muhie Yimam** and Chris Biemann. Demonstrating PAR4SEM – A Semantic Writing Aid with Adaptive Paraphrasing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 48–53, Brussels, Belgium, 2018. Online: http://aclweb.org/anthology/D18-2009

3. Gregor Wiedemann, **Seid Muhie Yimam**, and Chris Biemann. New/s/leak 2.0 – Multilingual Information Extraction and Visualization for Investigative Journalism. In *Proceedings of the 10th International Conference on Social Informatics (SocInfo 2018)*, St.Petersburg, Russia, 2018b. Online: https://arxiv.org/abs/1807.05151

4. **Seid Muhie Yimam** and Chris Biemann. Par4Sim – Adaptive Paraphrasing for Text Simplification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 331–342, Santa Fe, NM, USA, 2018. Online: http://aclweb.org/anthology/C18-1028

5. **Seid Muhie Yimam**, Sanja Štajner, Martin Riedl, and Chris Biemann. CWIG3G2 - Complex Word Identification Task across Three Text Genres and Two User Groups. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 401–407, Taipei, Taiwan, 2017c. Online: http://www.aclweb.org/anthology/I17-2068

6. **Seid Muhie Yimam**, Sanja Štajner, Martin Riedl, and Chris Biemann. Multilingual and Cross-Lingual Complex Word Identification. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 813–822, Varna, Bulgaria, 2017b. Online: https://doi.org/10.26615/978-954-452-049-6_104

7. **Seid Muhie Yimam**, Heiner Ulrich, Tatiana von Landesberger, Marcel Rosenbach, Michaela Regneri, Alexander Panchenko, Franziska Lehmann, Uli Fahrer, Chris Biemann, and Kathrin Ballweg. New/s/leak – Information Extraction and Visualization for Investigative Data Journalists. In *Proceedings of ACL-2016 System Demonstrations*, pages 163–168, Berlin, Germany, 2016c. Online: http://www.aclweb.org/anthology/P16-4028

8. **Seid Muhie Yimam**, Chris Biemann, Ljiljana Majnarić, Šefket Šabanović, and Andreas Holzinger. Interactive and Iterative Annotation for Biomedical Entity Recognition. In Yike Guo, Karl Friston, Faisal Aldo, Sean Hill, and Hanchuan Peng, editors, *Brain Informatics and Health*, pages 347–357, London, UK, 2015. ISBN 978-3-319-23344-4. Online: https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2015-yimametal-bih-london.pdf

9. Darina Benikova, **Seid Muhie Yimam**, Prabhakaran Santhanam, and Chris Biemann. GermaNER: Free Open German Named Entity Recognition Tool. In *Proceedings of GSCL 2015*, pages 31–28, Essen, Germany, 2015. Online: https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2015-benikovaetal-gscl2015-germa.pdf

10. **Seid Muhie Yimam**, Richard Eckart de Castilho, Iryna Gurevych, and Chris Biemann. Automatic Annotation Suggestions and Custom Annotation Layers in WebAnno. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. System Demonstrations*, pages 91–96, Baltimore, MD, USA, 2014. Online: http://www.aclweb.org/anthology/P14-5016

11. **Seid Muhie Yimam**, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations. In *Proceedings of ACL 2013 System Demonstrations*, pages 1–6, Sofia, Bulgaria, 2013. Online: http://www.aclweb.org/anthology/P13-4001.pdf

Workshop Proceedings

1. **Seid Muhie Yimam**, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. A Report on the Complex Word Identification Shared Task 2018. In *Proceedings of The 13th Workshop on Innovative Use of NLP for Building Educational Applications, NAACL 2018 Workshops*, pages 66–78, New Orleans, LA, USA, 2018. Online: http://aclweb.org/anthology/W18-0507

2. **Seid Muhie Yimam**, Steffen Remus, Alexander Panchenko, Andreas Holzinger, and Chris Biemann. Entity-Centric Information Access with Human in the Loop for the Biomedical Domain. In *Proceedings of the Biomedical NLP Workshop associated with RANLP 2017*, pages 42–48, Varna, Bulgaria, 2017a. Online: https://doi.org/10.26615/978-954-452-044-1_006

3. Martin Müller, Kathrin Ballweg, Tatiana von Landesberger, **Seid Muhie Yimam**, Uli Fahrer, Chris Biemann, Marcel Rosenbach, Michaela Regneri, and Heiner Ulrich. Guidance for Multi-Type Entity Graphs from Text Collections. In Michael Sedlmair and Christian Tominski, editors, *EuroVis Workshop on Visual Analytics (EuroVA)*, Barcelona, Spain, 2017. ISBN 978-3-03868-042-0. Online: https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2017-mueller-eurova-newsleak-guidance.pdf

4. Richard Eckart de Castilho, Éva Mújdricza-Maydt, **Seid Muhie Yimam**, Silvana Hartmann, Iryna Gurevych, Anette Frank, and Chris Biemann. A web-based tool for the integrated annotation of semantic and syntactic structures. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 76–84, Osaka, Japan, 2016. Online: http://aclweb.org/anthology/W16-4011

5. Kathrin Ballweg, Florian Zouhar, Patrick Wilhelmi-Dworski, Tatiana von Landesberger, Uli Fahrer, Alexander Panchenko, **Seid Muhie Yimam**, Chris Biemann, Michaela Regneri, and Heiner Ulrich. New/s/leak – A Tool for Visual Exploration of Large Text Document Collections in the Journalistic Domain. In *VIS conference 2016 collocated with the Workshop on Visualisation in Practice*, Baltimore, MD, USA, 2016. Online: http://www.aclweb.org/anthology/P16-4028

6. **Seid Muhie Yimam**, Héctor Martínez Alonso, Martin Riedl, and Chris Biemann. Learning Paraphrasing for Multiword Expressions. In *Proceedings of the 12th Workshop on Multiword Expressions*, pages 1–10, Berlin, Germany, 2016b. Online: http://www.aclweb.org/anthology/W16-1801

7. **Seid Muhie Yimam**. Narrowing the Loop: Integration of Resources and Linguistic Dataset Development with Interactive Machine Learning. In *Proceedings of HLT-NAACL: Student Research Workshop*, pages 88–95, Denver, CO, USA, 2015. Online: http://www.aclweb.org/anthology/N15-2012

8. Richard Eckart de Castilho, Chris Biemann, Iryna Gurevych, and **Seid Muhie Yimam**. WebAnno: a flexible, web-based annotation tool for CLARIN. In *CLARIN Annual Conference*, Soesterberg, The Netherlands, 2014. Online: https://www.clarin.eu/sites/default/files/cac2014_submission_6_0.pdf

9. Darina Benikova, Uli Fahrer, Alexander Gabriel, Manuel Kaufmann, **Seid Muhie Yimam**, Tatiana von Landesberger, and Chris Biemann. Network of the Day: Aggregating and Visualizing Entity Networks from Online Sources. In *Proceedings of NLP4CMC at KONVENS2014*, pages 48–52, Hildesheim, Germany, 2014b. Online: https://hildok.bsz-bw.de/files/277/01_07.pdf

Eidesstattliche Versicherung
Declaration on oath

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, on oath, that I have written the present dissertation by my own and have not used other than the acknowledged resources and aids.

Hamburg, den *Unterschrift*

Adaptive Approaches to Natural Language Processing in Annotation and Application

Zusammenfassung

Die meisten Anwendungen im Bereich Sprachverarbeitung (Natural Language Processing, NLP) basieren auf Komponenten des Maschinellen Lernens. Die meisten dieser Komponenten müssen in einem überwachten Setting trainiert werden und benötigen hierfür ausreichend Trainingsdaten, die von Domänen-Experten manuell gelabelt oder annotiert werden müssen. Die Erstellung eines solchen Datensatzes bietet viele Herausforderungen: zunächst sollen die Domänen-Experten 1) wissen, was zu annotieren ist und wie man annotiert (Erstellung von Annotations-Guidelines), 2) prüfen, ob die existierenden Annotations-Tools ausreichend sind, oder ein neues Annotations-Tool erstellen, und 3) die Qualität der annotierten Daten überprüfen. Zweitens ist das Annotieren kostenaufwendig, vor allem um die Annotatoren zu bezahlen. Drittens dauert das Annotieren sehr lange, wenn man viele Daten sammeln will. Zuletzt können die gesammelten Daten auch veralten, wenn die Anforderungen an die NLP-Anwendung sich mit der Zeit ändern. Das nennt sich **semantische** oder **Konzept-Verschiebung**.

Diese Doktorarbeit will drei Ziele erreichen: **die Erstellung von schnellen Annotations-Tools**, **die Erstellung und Integration von Sprachressourcen für NLP-Anwendungen** und **die Integration von adaptiven Modellen in NLP-Anwendungen**. Der Ansatz des schnellen Annotierens (rapid annotation) setzt den Akzent auf die schnelle Erledigung des Annotations-Tasks, indem verschiedene Strategien direkt im Tool implementiert werden. Um eine adaptive und personalisierte NLP-Anwendung zu entwickeln, bauen wir ein adaptives maschinell lernendes Modell in die Anwendung ein, das durch die Benutzung ständig lernen und das Modell aktualisieren kann.

Die Doktorarbeit besteht aus fünf Teilen. Teil I beschreibt die Problemstellung an sich, wissenschaftliche Probleme und den Hintergrund dieser Studie, Teil II stellt die Annotationen und die NLP-Tools vor, die wir in der Arbeit an dieser Doktorarbeit entwickelt haben. Teil III erörtert die Ansätze, die für das Sammeln von NLP-Ressourcen mit Semantik-Kenntnissen benutzt werden, und Teil IV beschreibt die Versuchsaufbauten und die Ergebnisse. Zuletzt schließt Teil V mit den wichtigsten Ergebnissen und Erkenntnissen aus dieser Doktorarbeit ab.

Im **Teil I** stellen wir die Hauptziele der Doktorarbeit vor (siehe Kapitel 1) und diskutieren die Grundzüge des schnellen Annotierens und der adaptiven NLP-Technologien (siehe Kapitel 2).

Teil II dieser Doktorarbeit stellt die verschiedenen Tools vor, die wir für die schnelle, adaptive und personalisierte Annotation entwickelt haben. Im Kapitel 3 erläutern wir das Design und die Implementierung des online Annotations-Tools **WebAnno**. WebAnno erfüllt verschiedene Anforderungen und Eigenschaften des schnellen Annotierens, wie die Unterstützung von Annotations-Korrekturen, Automation um Annotationsvorschläge zu generieren und eingebautes Bewerten von Annotationsentscheidungen. Außerdem haben wir WebAnno mit verschiedenen Komponenten in der Benutzeroberfläche ausgestattet, die schnelles Annotieren erleichtern sollen.

Im Kapitel 4 stellen wir **Par4Sem** vor, eine semantische Schreibhilfe, welche die Erstellung von Dokumenten mit einer personalisierten Paraphrasen-Komponente erleichtert. Das Tool ahmt ein Textverarbeitungssystem nach, aber es nutzt ein integriertes adaptives Modell, das lexikalisches Paraphrasieren anbietet. Im Kapitel 5 stellen wir verschiedene Informations-Visualisierungstools (**InfoVis**) vor, die für das Sammeln von Trainings-Daten benutzt werden können und Teil von Daten-Visualisierungen und Erkundung sind.

Im **Teil III** dieser Doktorarbeit erörtern wir die Datensammlung, Versuchsaufbauten und die Ergebnisse, die wir für semantische NLP-Anwendungen gesammelt haben. Im Kapitel 6 untersuchen wir den Einfluss des Kontextes auf die Paraphrasierung. Paraphrasierung ist ein Ansatz um Texte umzuschreiben und dabei äquivalente Texte mit dem gleichen Informationsgehalt zu generieren. Für dieses Experiment erstellen wir aus verschiedenen Paraphrasen-Ressourcen Paraphrase-Kandidaten für Zielwörter und Phrasen und präsentieren diese Kandidaten mehreren Crowdworkern, damit diese die Kandidaten im gegebenen Kontext nach Reihenfolge ordnen können. Die erzielten Ergebnisse zeigen, dass der Kontext-Bezug das Paraphrasieren verbessert.

Im Kapitel 7 untersuchen wir, ob Muttersprachler andere Anforderungen an Textvereinfachung stellen als Nicht-Muttersprachler. Dazu führen wir eine Annotationsaufgabe durch, der komplexe Wörter und Phrasen ermittelt (complex word identification, CWI). Das Ziel dabei ist es, Teile eines Texts auszuzeichnen, die schwierig zu verstehen sein können. Dann prüfen wir, ob Modelle des maschinellen Lernens, die auf einer Sprache trainiert wurden, mittels de-

lexikalisierten Features dazu genutzt werden können, komplexe Phrasen vorauszusagen für 1) verschiedene Nutzergruppen (Muttersprachler und Nicht-Muttersprachler), 2) verschiedene Textgattungen (Wikipedia und Nachrichtentexte) und 3) verschiedene Sprachen (ein auf Englisch trainiertes CWI-Modell wird für die Erkennung komplexer Phrasen im Deutschen verwendet). Die wichtigste Erkenntnis ist, dass es möglich ist, ein CWI-System für eine Sprache zu entwickeln und dessen Modell für die Erkennung von komplexen Wörtern und Phrasen einer anderen Sprache zu nutzen.

Im **Teil IV** präsentieren wir die experimentellen Ergebnisse verschiedener Arten von Annotationsautomation und adaptiver NLP-Anwendungen. Im Kapitel 8 führen wir verschiedene Experimente für schnelles und automatisiertes Annotieren mittels des WebAnno-Tools durch und evaluieren diese. Bezüglich der Annotations-Zeit mit der Automation-Komponente von WebAnno, zeigen wir, dass der Ansatz des schnellen Annotierens den Annotations-Vorgang bei einer sequentiellen Annotationsaufgabe um den Faktor 3 bis 4 beschleunigt. In einem Experiment für das Erkennen von biomedizinischen Entitäten (biomedical Named Entity Recognition), ermöglicht die adaptive Komponente von WebAnno eine schnelle und einfache Annotation von medizinischen Entitäten, wo das Modell bereits mögliche Entitäten vorschlägt, nachdem es anhand von wenigen annotierten medizinischen Abstracts mitgelernt hat.

Im Kapitel 9 diskutieren wir die Ergebnisse und Folgerungen der Experimente zu adaptiven und personalisierten NLP-Anwendungen. Eine adaptive und personalisierte NLP-Anwendung wird durch die Integration von adaptiven Modellen des Maschinellen Lernens in einer semantische Schreibhilfe realisiert (**Par4Sem**). Das erste adaptive Modell ist eine Komponente zur Erkennung von komplexen Wörtern, die adaptiv komplexe oder schwierige Wörter durch Benutzerinteraktion lernt. Wir instanziieren das adaptive Modell durch ein Baseline-System der CWI-Datensätze. Die zweite Komponente ist ein Paraphrasen-Ranking-Modell, das adaptiv lernt, wie man die Paraphrasen-Kandidaten ordnet. Wir führen die Experimente mit der Crowdsourcing Plattform Amazon Mechanical Turk (MTurk) durch, wo Par4Sem explizit dafür verwendet wird, Texte für eine bestimmte Nutzergruppe zu vereinfachen (**Kinder**, **Sprachenlernende** oder **Menschen mit Lesebeeinträchtigungen**). Um die Anpassungsfähigkeit des Modells zu testen, führen wir ein Experiment mit 9 Interationen durch und aktualisieren das Modell nach jeder Iteration. Die Ergebnisse des Paraphrasen-Rankings anhand der Ranking-Metrik NDCG (normalized discounted cumulative gain) zeigen, dass die Performanz sich substantiell

verbessert.

Abschließend zeigt sich, dass bei der schnellen Fortentwicklung der Strategien des Maschinellen Lernens die Integration adaptiver Modelle in einer Anwendung mit selbst-aktualisierenden Fähigkeiten Vorteile bietet. Zudem wird ein generisches Modell in NLP-Anwendungen und Systemen, die zunehmend auf Maschinelles Lernen setzen, die Erfüllung der Anforderungen an solche Anwendungen beschränken. Anstatt viel Energie darauf zu verwenden, Trainingsdaten zu sammeln und ein Modell zu trainieren und anzupassen, das möglicherweise schnell überholt ist, sollten wir unseren Fokus auf adaptive und personalisierte Modelle setzen, die beständig durch Benutzerinteraktion lernen können.

Adaptive Approaches to Natural Language Processing in Annotation and Application

Abstract

Most applications in natural language processing (NLP) are based on machine learning components. Most of these components need to be trained in a supervised way and require a substantial amount of training data. These training data have to be manually labeled or annotated by domain experts. Creating such dataset is a challenging task: first, domain experts 1) need to know what to annotate and how to annotate (compiling an annotation guideline), 2) study if existing annotation tools are adequate or develop a new annotation tool, and 3) assess the quality of annotated data. Second, the annotation task is expensive mainly to compensate the annotators. Third, the annotation task takes much time to collect enough data. Finally, the collected data might get obsolete if the requirement of the target NLP application changes over time, which is known as **concept drift** or **semantic drift**.

This work tackles three main questions: **how to build rapid annotation tools**, **how to build and integrate resources to NLP applications**, and **how to integrate adaptive models into NLP applications**. A rapid annotation approach focuses on completing the task much quicker by employing different strategies within the annotation tool. For the development of an adaptive and personalized NLP application, we embed an adaptive machine learning model into the application that can continuously learn and update its model from usage data.

The thesis comprises of five parts. Part I describes the problem statements, research problems, and background of the study, Part II presents the annotation and NLP tools we have developed during the thesis work, Part III discusses the approaches used to collect semantic-aware NLP resources, Part IV describes the experimental setups and results, and finally Part V concludes with main findings and results of the thesis.

In **Part I**, we present the main goals of the thesis (see Chapter 1) and discuss the foundations of the rapid annotations and adaptive NLP technologies (see Chapter 2).

Part II of the thesis presents the different tools we design for rapid, adaptive, and personalized annotations. This part describes the technological basis, which needs to be especially

robust because of the user-interfacing nature of the experiments. In Chapter 3, we discuss the design and implementation approaches for the development of the **WebAnno** online annotation tool. WebAnno incorporates different rapid annotation properties such as support of annotation correction, automation to produce annotation suggestions, and built-in annotation adjudication. We also extend WebAnno with different user interface components that are meant to facilitate rapid annotation.

In Chapter 4, we present **Par4Sem**, a semantic-aware writing aid tool that enhances document composing with an adaptive and personalized paraphrasing component. The tool mimics a standard word processor, but it has an integrated adaptive model that provides semantic-aware text paraphrasing capability. In Chapter 5, we present the different information visualization (**InfoVis**) tools, which can be used to collect training data as part of data visualization and exploration.

In **Part III** of the thesis, we discuss data collection, experimental setups, and results obtained particularly for semantic-aware NLP applications. In Chapter 6, we investigate the impact of context for the paraphrasing tasks. Paraphrasing is an approach of re-writing texts to produce equivalent texts that convey the same information. For this experiment, we produce candidate paraphrases for target words or phrases from different paraphrase resources and present the candidates for multiple crowdsourcing workers to re-rank candidates based on their context. The results obtained show that awareness of context improves paraphrase ranking.

In Chapter 7, we investigate whether native and non-native language speakers have different demands for text simplification or not. We first conduct complex words or phrases identification (CWI) annotation task, where the goal is to determine parts of a text that could pose difficulty to understand a text. Then, we investigate if machine learning models build for one language, using de-lexicalized features, can be used to predict complex phrases for 1) different user groups (native and non-native users), 2) different text genres (Wikipedia and news articles), and 3) different languages (a CWI model trained for English can be used to identify complex phrases for German). The most important finding is that it is possible to build a CWI system for one language and use the model to identify complex words or phrases for other languages.

Part IV contains the main experimental results for the setups of annotation automation and an adaptive NLP application. In Chapter 8, we conduct and evaluate different rapid and automation annotation experiments using the WebAnno annotation tool. Regarding the anno-

tation time using the WebAnno automation component, we demonstrate that the rapid annotation approach speeds up the annotation process by the factor of 3 to 4 on sequence tagging tasks. In an experiment with biomedical named entity recognition, the adaptive component of WebAnno allows for fast and easy annotation of medical entities where the model suggests useful entities already after the annotation of a handful of medical abstracts.

In Chapter 9, we discuss the results and implications of the experiments on adaptive and personalized NLP application. An adaptive and personalized NLP application is realized by integrating adaptive machine learning models into a semantic writing aid tool (**Par4Sem**). The first adaptive model is a complex word identification component, which adaptively learns the complex or difficult words from the user interaction. We instantiate the adaptive model using a baseline system of the CWI datasets. The second component is a paraphrase ranking model, which learns to rank or order candidate paraphrases adaptively. We experiment using the Amazon Mechanical Turk (MTurk) crowdsourcing where Par4Sem is explicitly used to simplify texts for a given target reader (**children**, **language learners**, or **people with reading impairment**). To test the adaptability of the models, we run the experiments for 9 iterations where the models are updated for each iteration. The experimental results for the paraphrase ranking show that in every iteration, there is a substantial increase in performance based on the normalized discounted cumulative gain (NDCG) learning to rank evaluation metrics.

In conclusion, with the rapid advancement of machine learning strategies, the integration of adaptive models into the application with a self-updating capability is a way forward. Moreover, as machine learning models are becoming prevalent in many NLP applications and software systems, a generic model will have limitations in fulfilling the target application's requirement. Instead of investing even more efforts in collecting training datasets and training a static machine learning model, we should focus on building an adaptive and personalized model that is capable of learning continuously from the user interaction.

Contents

Acknowledgments

Alhamdulillah: praise be to God, the Almighty, for giving me this noble opportunity and helping me to complete my thesis.

My academic path was very long and it is difficult to travel this much without supports at every point in time. The chance that I could not go to even an elementary school, late alone a university to attain Ph.D. was very likely. A page or two will not be enough to mention all who helped in my journey. To those of you I have missed your name here, you are not forgotten and your contribution is no less important by any means. A sincere thank you goes to all of you.

A special thank you goes to my supervisor, Chris Biemann, you allowed me to work on such outstanding scientific projects as a software developer, understood my everlasting interest to pursue my Ph.D., and finally allowed me to start the journey. I "dreamed bright" and you helped me "succeed"! I have got tremendous and unconditional supports throughout this work from you, you pointed me to directions that otherwise could not be imagined, and guided me with your special skills of "personalized" supervision. Thank you very much.

I am grateful to my colleagues at LT. Be it academic or social issues, you are always beside me to lend your hand without hesitation. What a fantastic group to work with, a perfect balance between work and fun and coffee!

A lot of people help in reading and commenting on my final draft of the thesis. Your comments were useful in re-shaping my thesis in this current form. Thank you all of you, Rawda Asefa, Steffen Remus, Martin Riedl, Darina Gold, Wolfgang Menzel, Chris Biemann, Özge Alaçam and many more.

My research visit to the University of Copenhagen and the collaboration with Héctor Martínez Alonso was the foundation to start crowdsourcing experiments. Héctor was very helpful in

shaping my thesis idea and providing theoretical guidelines. Thank you Héctor for your encouragement and the timely feedback.

I am grateful to Richard Eckart de Castilho for his technical support during my stay at UKP lab, TU Darmstadt. You helped me to learn the basic technologies for the development of the WebAnno tool.

I am also grateful to my teachers throughout the ladder. Especially, my elementary school teachers in "Doyle Elementary school", a remote school in Ethiopia, you have planted in me the special desire of education. Even in the absence of basic needs for life, you dare to teach and shape students in the best possible way, you are my heroes.

For all of my publications, I have obtained enormous comments from the NLP communities. Thank you all for the useful comments that finally are used to produce this thesis. I also thank Rawda Asefa and Sisay Adugna for their help in proofreading the translation of our COLING 2018 paper abstract into Amharic.

I am also grateful to different organizations for conference travel and publication supports. Parts of this work have been partially supported by the SEMSCH project at the University of Hamburg, funded by the German Research Foundation (DFG) and the German BMBF grant to the CLARIN-D project.

To my late father Muhie Yimam, being a soldier you clearly understand what education meant and how important it is in life. I remember how eager you were to send me school, accompanying me to the farthest school in the middle of the night (as I were too young to go alone) to attend my secondary school. I wish you were here today to witness how far I traveled due to your encouragement.

A very special thank you goes out to my mother Alemnesh Yimer. When my father was on duty, you provide everything you can to send me school. When my father was passed away, you assumed every responsibility instead of letting your boy drop from school out. I am proud of you!

And finally, my wife, Nefisa Kedir, and my children, Yusra, Husna, and Mohamed, this thesis is a cumulative one. It would be not possible to accomplish it if it was not for your day-to-day support and understanding. Thank you very much!

Listing of figures

List of Tables

Glossary

Adaption	The word adaption refers to the nature of an application, for example, an annotation tool or an NLP application, where it learns from the user interaction towards a given goal. For example, if a user is annotating mentions of medical entities in a text, the model tries to learn to identify such terms based on previous examples automatically. Note that the word adaption is used throughout the thesis over the commonly used word called **adaptation**
Adaptive annotation	An annotation process that supports **adaption**.
Adaptive annotation tool	An annotation tool that allows to conduct **adaptive annotation**.
Adaptive NLP application	An NLP application that integrates an **adaptive annotation** component.
Annotation target	An annotation target is the part of a document that is going to be annotated. For example, for POS tag annotation, the tokens words are the annotation targets.
AnonML	AnonML is a web-based tool that we have developed in the scope of this work, which helps in anonymizing legal documents.
Auto-forwarding	Auto-forwarding is a rapid annotation behavior in WebAnno, where the annotation selection/highlighting pointer automatically moves to the next token or word once the user completes an annotation on the current token.

Automatic suggestions	The ability to produce **suggestions** automatically based on a machine learning model
Candidate paraphrases	List of of phrases or words that are equivalent in meaning for a given target word or phrase.
Complex phrase	see Complex word.
Complex word	We will use complex phrase, complex word, or complex units interchangeably, which are parts of a text that could pose difficulty for understanding a given text.
Crowdsourcing	Regarding annotation problem, it is a form of collecting annotations quickly from a crowd (workers over the Internet) with less cost.
Dataset	The term dataset might be used interchangeably with training examples, usage dataset, labeled dataset, annotated dataset, and training dataset, which is a resource that is used to train and evaluate a machine learning model.
Delexicalized feature	A machine learning feature, which does not depend on the actual lexical word or term but its characteristics such as the length of the word or the number of vowels in the word.
Document anonymization	Document anonymization is the process of replacing sensitive data with an arbitrary representation to ensure the confidentiality of sensitive data.
Gazetteers	Gazetteers are manually compiled list of mentions or entities for a given domain. For example, named entity recognition can use compiled lists of person or location names to detect entities in texts automatically.
Human intelligence task	A "single" task that is going to be completed by one crowdsourcing worker (hence collecting training examples) so that a reward can be granted.

Human-in-the-loop	The phrase human-in-the-loop refers to two concepts. For rapid annotation, it is the possibility that the annotation tool allows re-training a model by the annotator (the **human**) during the annotation process (the **loop**) in order to produce better suggestions.
Key-binding	Key-binding is a rapid annotation behavior in WebAnno, which is to bind a tag a key on a keyboard as a shortcut to annotate a token.
Learning to rank	It is a supervised machine learning approach that is trained mainly to discover the best order for a list of items based on features extracted from each item.
Multi-word expressions	Multi-word expressions (MWEs) are expressions or phrases, composed of 2 or more words, which are typically used to express a specific concept. Examples: **by and large**, **fresh air**, **New York**, **made in**, ...
Network of the day	Network of the day is a tool for visualization of entities and their relations from daily news extract that we have developed in the scope of this work. It helps to collect training data beside data visualization and exploration.
New/s/leak	New/s/leak is a visualization tool for data journalists that we have developed in the scope of this work. It helps to collect training data beside data visualization and exploration.
Par4Sem	Par4Sem is a semantic-aware writing aid tool that provides adaptive paraphrasing support for document composing that we have developed in the scope of this work. It forms the main technology basis for the adaptive NLP experiments.
Paraphrase resources	It is an NLP resource, where a source word or phrase is connected to a target word or phrase that is identical in meaning.

Paraphrase target	The word or phrase in a document that is going to be paraphrased is called a target or paraphrase target.
Personalized	In this thesis, the term **personalized** is mainly referring to the behavior of a machine learning model, which adapts towards the preference of the user.
Qualification requirement	For crowdsourcing annotation, it is a way of filtering workers, for example based on their previous annotation performance.
Rapid annotation	Rapid annotation, in general, refers to an approach that helps the collection or annotation of training data using an annotation tool in order to complete an annotation task faster without affecting the quality of the dataset. It extends from simple user interface design that makes annotation creation faster and simpler to an automatic and adaptive machine learning approach that produces suggestions, which can be corrected by the user.
Rapid annotation tool	An annotation tool that supports **rapid annotation**.
Repeat annotation	Repeat annotation is a rapid annotation behavior in WebAnno, which allows repeating annotation of similar words or phrases for all documents in a project that appear as a suggestion. The annotator can accept or reject the suggestions in the suggestion view of the automation page.
Suggestion view	Suggestion view or suggestion pane is an annotation component in WebAnno that displays suggestions from an internal machine learning model. The same view is used to display suggestions from an externally annotated documents where users are required to correct the annotations.
Suggestions	In annotation tool, the ability to identify **annotation targets** and providing or suggesting a possible label is called suggestion

Tag-set	A collection of tags.
Tags	Labels that are used to indicate the category a given annotation unit (example tokens, phrases). For example, POS tags are used to indicate the word category.
Text simplification	Generally, text simplification is an NLP task where a given text corpus is modified (both in syntactic and lexical form) to make its content understandable or readable for a target reader.
Training data	see Dataset
Training examples	see Dataset
Usage data	Dataset or training examples collected while using an NLP application. see also Dataset
WebAnno	WebAnno is a web-based, distributed, and generic rapid annotation tool that we have developed in the scope of this work. It forms the main technology basis for the rapid annotation experiments.
Writing aid	Writing aid is an integrated component to text editors that help the writer in composing texts. In the context of this thesis, writing aid, more specifically semantic writing aid, is a tool that supports document writing by providing paraphrases in context.

List of Abbreviations

B-CLL	B-chronic lymphocytic leukemia
BioNLP	Biomedical natural language processing
BRAT	brat rapid annotation tool
CLARIN	Common language resources and technology infrastructure
CoNLL	Computational natural language learning (Conference)
CWI	Complex words or phrases identification
D-SPIN	Deutsche Sprachressourcen-Infrastruktur
F-score	Also F-1 score or F-measure, it is the harmonic average of the precision and recall.
FoLiA	Format for linguistic annotation
GrAF	The graph annotation format
GUI	Graphical user interface
HIT	Human intelligence task
IAA	Inter annotator agreement
InfoVis	Information visualization

LAF	Linguistic annotation framework
MAP	Mean average precision
MEDLINE	Medical literature analysis and retrieval system online
MTurk	The Amazon Mechanical Turk
MWE	Multi-word expression
NDCG	Normalized discounted cumulative gain
NER	Named entity recognition
NLP	Natural language processing
POS	Part of speech
PPDB	The Paraphrase Database
RankLib	It is a library of learning to rank algorithms.
REST	Representational state transfer
TCF	Text corpus format
TEI	Text encoding initiative
TSV	Tab separated value
UI	User interface
UMLS	Unified medical language system
XML	Extensible markup language

Part I

Introduction and Background of Annotation and Adaption

ቀስ በቀስ እንቁላል በእግሩ ይሄዳል!
Over time, an egg starts walking on foot!
an Ethiopian Quote

1 Introduction

Computers are becoming remarkably effective in solving complex problems in computational and artificial intelligence areas like playing chess games (David et al., 2016), winning the jeopardy question and answer (Ferrucci et al., 2010), beating the world's champion in the game of Go with Google's AlphaGo (Silver et al., 2016), and determining best routes for a self-driving car (Chen and Huang, 2017). However, it is still an unsolved problem, when it comes to understanding and processing even a straightforward command in human language. It is still challenging to design systems that can process language related artifacts the way humans do.

The research field of natural language processing (NLP) mainly deals with processing human or natural languages (both text and speech), in order to easily process and understand language data using computers. NLP is one of the disciplines in computer science and engineering, which focuses in understanding human languages by computers, based on approaches and guidelines from computational linguistics (Tsujii, 2011).

Evidence suggests that NLP is one of the core components and principal initiators of the cur-

rent artificial intelligence (Nilsson, 2009). To process and understand natural languages using computers, it is required to provide examples or annotations where it can learn patterns from the data. NLP applications usually integrate a machine learning component that requires annotated data, where it learns from the labeled annotation in order to apply the rules to unseen texts or unlabeled data. In general, the development of NLP application is realized either with **rule-based** or **machine learning** approaches (Crowston et al., 2010; Pilán et al., 2014).

In the rule-based, also known as **knowledge-based** or **pattern-based** approaches, the development of NLP application requires the compilation of rules or patterns that will fit the problem at hand. The rules can be regular expressions, dictionaries, or syntactic structures that can be easily constructed from a given data. The main problems in rule-based NLP application development are 1) the development of rules or knowledge-based resources is tedious and 2) the rules developed will not be generic as they are developed specifically to a given problem for a particular language concerned.

However, most NLP applications are developed based on different machine learning components instead of handcrafted rules. This approach is more adaptable and works moderately well for different NLP applications, but it requires a large number of training datasets from which the algorithm draws language-related patterns during prediction (Pustejovsky and Stubbs, 2012). The development or collection of such training examples is also laborious, expensive, and expects a certain degree of human expertise in the area of NLP application during the annotation process.

Moreover, the collection of training examples mainly faces two major challenges: 1) it usually does not scale very well since annotation of a large dataset is extremely expensive, and 2) the need of the application might change over time where a static model trained on the datasets collected at a particular time fails to serve its purpose. The second issue is commonly known as **concept drift** or **semantic drift** (Hoens and Chawla, 2012; Kulesza et al., 2014; Tsymbal, 2004; Žliobaitė et al., 2016).

If a machine learning model is the primary component of NLP applications, and we know that getting training examples is a challenging task, we need to investigate 1) how to collect the training examples **rapidly**, and 2) how to avoid the traditional, protracted, and costly way of collecting training examples as much as possible, and replace the annotation process with an **embedded** model into the target NLP application, which relies on **usage data**.

Rapid annotation has different aspects. The primary purpose of a rapid annotation paradigm is to complete the annotation process much faster than it is done traditionally, without compromising the quality of the training data. Rapid annotation can be accomplished by developing an annotation tool that is easy to use and install, supports a quicker annotation process, and provides annotation suggestions and correction capability.

In this thesis, we first investigate the main problems and limitations of the existing annotation processes and overcome them using a rapid annotation paradigm. We demonstrate our rapid annotation approach using **WebAnno** and other annotation and visualization tools, which we develop during the thesis work. Most significantly, WebAnno is a rapid annotation tool, which is designed and implemented after investigating the problems with the existing annotation tools.

The second primary goal of this thesis goes beyond the development of rapid annotation tools. It focuses on the integration of an **adaptive** and **personalized** machine learning model into an NLP application that depends on **usage data** to build its model. More particularly, the text simplification NLP application is targeted to demonstrate the adaptability, personalizability, and usability of the embedded model, using the **Par4Sem** semantic writing aid tool. Par4Sem is a semantic writing aid tool, which aims to enhance document writing with a semantic-aware paraphrasing capability, mainly by providing alternative paraphrases for words or phrases in a text.

1.1 Rapid Annotation, Human-in-the-loop, Adaptive and Personalized NLP Applications

The annotation process is usually cumbersome, which takes too much effort concerning time and money. Consider for example Figure 1.1.1, which shows the traditional pipeline in building a training dataset for a given NLP application. It starts with studying the annotation requirements, designing the annotation tool, and developing the annotation guidelines. Once the annotation process is started, the NLP engineer should evaluate the quality of the collected training data before deciding to stop the annotation process. The process iterates a few times until the expected amount of training data are collected. At the end of the annotation process, the final training data is compiled, a machine learning model is built, and the model is integrated into a target NLP application. From Figure 1.1.1 (marked by X), we can see that there is

no direct feedback between the target application and the annotation process at all.

The primary objective of this thesis is to refine the traditional annotation process in many aspects.

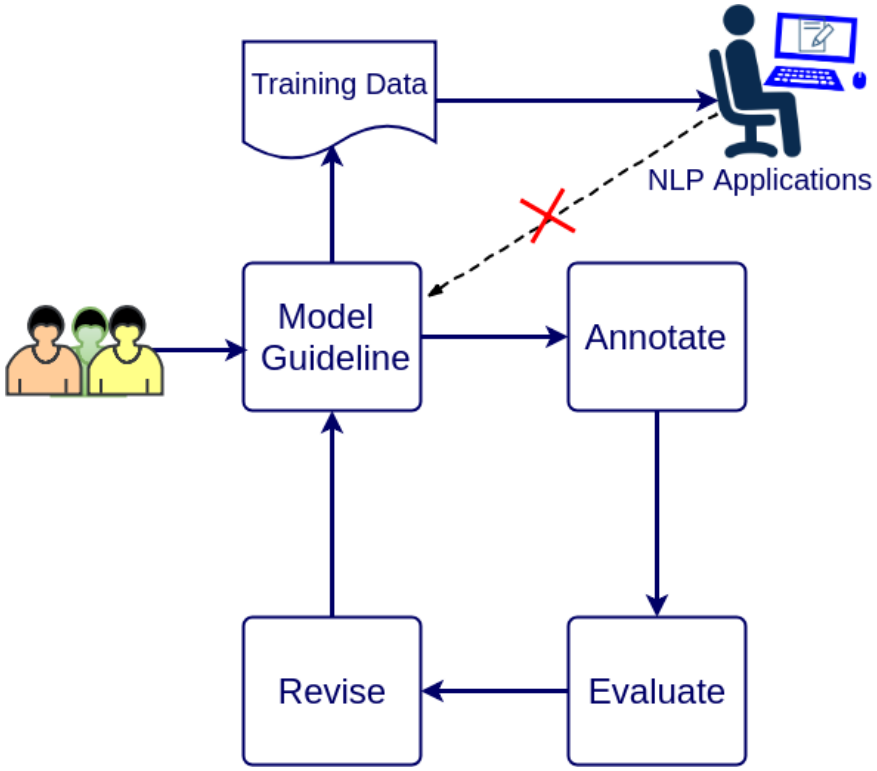


Figure 1.1.1: The traditional process of building a training dataset for a given NLP application. The annotation process repeats a few times until the required amount of training data is collected.

Overall, a **rapid annotation** process, as we discuss in Chapter 2 in detail, focuses on making the annotation process faster, easier, and cheaper. It could be attained by 1) making the annotation tool run in web and mobile applications instead of as a standalone desktop application, 2) enabling to annotate broader varieties of annotation types, supporting different file encodings, or allowing annotation for various languages at a time, 3) supporting the annotator in the annotation process by providing helpful hints and annotation guidelines, and 4) providing automatic suggestions of annotations where the annotator can accept or correct the suggestions.

Figure 1.1.2 depicts the general approach we followed in designing the rapid annotation paradigm,

both for adaptive annotation tools and personalized NLP applications. The rapid annotation tool differs from the traditional annotation process as it gets feedback immediately during annotation time (using the **human-in-the-loop** approach). The main objective is to make the annotation process much faster, avoiding the **annotate** → **evaluate** → **revise** → **annotate** iteration, which is usually performed outside of the annotation tool.

For the development of an adaptive and personalized NLP application, we entirely avoid the need to use annotation tools (marked by X in Figure 1.1.2) to collect training data. Instead, we embed the machine learning model inside the NLP application, which relies on **usage data** to train and update its model.

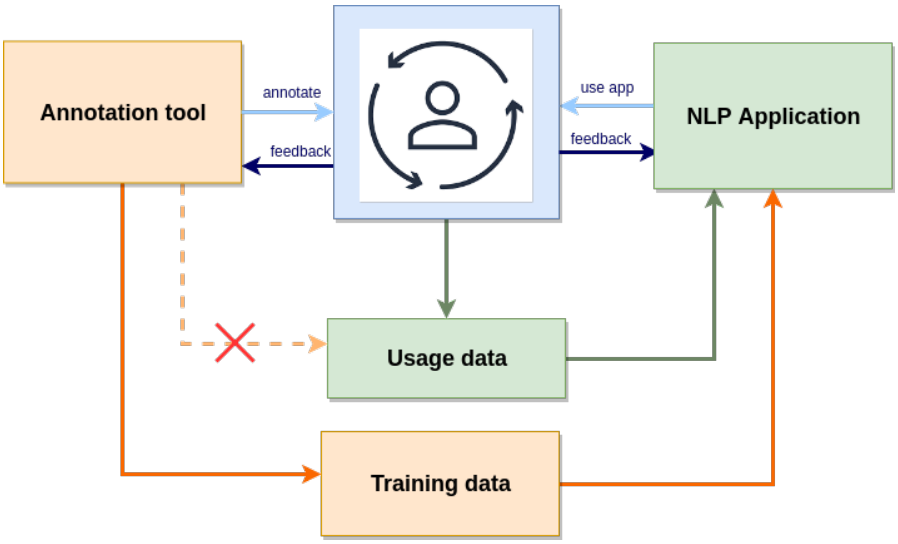


Figure 1.1.2: The rapid annotation process and the adaptive/personalized NLP application approach. For the rapid annotation process, the feedback from the user helps to train a machine learning model, which in turn provides suggestions. For the adaptive and personalized NLP application, the application depends on usage data to train its machine learning model.

1.2 Research Questions

This thesis broadly addresses two research problems: **How could a machine learn from usage data (user feedback) without explicitly obtaining training examples in advance?** and

How could a machine periodically adapt its model for an adaptive and personalized NLP application?

More specifically, three directions of research problems (RPs) are mainly addressed.

1. **RP1**: Tools for rapid annotation: In this research direction, we target to answer questions such as "**How can we develop annotation tools with rapid annotation characteristics?**" and "**what characterizes rapid annotation tools**?"

2. **RP2**: Resources for rapid annotation: In this direction of research, the focus is on "**How to build resources for NLP applications using rapid annotation tools?**" and "**How to integrate existing resources for adaptive NLP applications?**"

3. **RP3**: Rapid annotation with no-tools and no-resources: In this research direction, we focus on questions about "**How can NLP applications get training data from application usage without employing annotation tools?**" and "**How to build personalized machine learning models for an NLP application using the human-in-the-loop paradigm?**"

1.3 Overview of Methodologies

To accomplish the rapid, adaptive, and personalized annotation paradigm, we employ different approaches and methodologies.

Firstly, we identify the main bottlenecks for rapid annotations in the existing annotation tools. In this regard, we carry out different experiments and incorporate rapid annotation functionalities for annotation tools. Some of the rapid annotation contributions include designing an easy to use interface and implementing more transparent and seamless annotation interactions.

Secondly, we employ an adaptive machine learning model that can be integrated into annotation tools to provide annotation suggestions. Suggestion helps the annotator in many ways: 1) When the suggestion is correct, it considerably reduces the need to annotate manually, that is, annotators can only accept and proceed to the next annotation task. 2) Even if the suggestions are wrong, they already provide enough hints to the annotator by automatically identifying the

location of the annotation units, specifically for span based annotation types. The automatic selection of annotations for suggestion helps the annotator not to read the whole text in advance, and it also allows to correct the suggestions. 3) So finally, it allows the model to get better in performance through usage, which can be further used to automatically annotate the remaining text once the model is stable.

Thirdly, for NLP applications, an adaptive and personalized machine learning model can be integrated without employing an external annotation tool to collect the training data. In this approach, an empty model is integrated into the NLP application, particularly for a semantic writing aid tool, which depends on usage data to train and update the model over time.

For the first and second approaches, we develop an annotation tool (WebAnno) and visualization tools (such as New/s/leak and Network of the day) that incorporate the rapid annotation functions. We then use the tools to assess the effectiveness of the rapid annotation process. We discuss the details in Chapter 3, Chapter 5, and Chapter 8.

For the third approach, we develop an NLP application, more specifically a semantic-aware writing aid tool that updates its semantic-aware paraphrasing model from usage data. We experiment by hosting the tool into a crowdsourcing platform where paid workers used the tool to simplify texts (re-writing a given text in a simpler form for particular readers). We discuss the design and development of the tool in Chapter 4 while we present the experimental setups and results obtained in Chapter 9.

1.4 Contributions of the Work

In this sub-section, we briefly explain most of the contributions of the thesis. Details of the research problems, experimental setups, developed tools and resources, and the outcomes of the different experiments are detailed in the remaining chapters of the thesis. Note that the thesis is based on different experimental results and tools, previously described in a range of conference papers, articles, and journal papers.

1.4.1 Annotation, Visualization, and Adaptive NLP Tools

Towards the exploration of rapid annotation process, we investigate the main challenges with existing tools, and we develop different annotation and visualization tools addressing those

challenges. Some of the limitations of the existing annotation tools are 1) they are designed for specific annotation types or layers, 2) they are developed only for specific language, 3) they are not easy to install and use, and 4) they do not support multiple annotators. One of the rapid annotation tools we develop is **WebAnno**, which is a web-based, distributed, multi-user support annotation tool with advanced functionalities beyond annotation such as curation, automation, correction, and annotation progress management.

We explicitly design annotation tools to collect datasets to train and test machine learning models for specific linguistic or NLP applications. Another approach to collect a dataset is to use visualization tools built for different tasks than collecting training dataset. The primary purpose of visualization tools is to present a given dataset graphically to enable easy and fast data exploration. **New/s/leak** and **Network of the day** are some of the tools we develop specifically for the objectives of data exploration, investigation, and analysis. These tools integrate indirect capabilities to collect a dataset that can be used to enhance the machine learning model.

In addition to standalone annotation tools and visualization tools, we also develop an adaptive and personalized NLP application called **Par4Sem**, which is used to provide semantic-aware assistance for writing. The tool has an embedded machine learning model that can learn from user feedback (usage data). The model learns and provides predictions continuously, which we call it a **life-time-learning** model.

The following publications are related to annotation and visualization tools:

1. **Seid Muhie Yimam**, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations. In *Proceedings of ACL 2013 System Demonstrations*, pages 1–6, Sofia, Bulgaria, 2013. Online: http://www.aclweb.org/anthology/P13-4001.pdf

    - I was the main developer of the WebAnno code under supervision of the other authors, which acted as technical or scientific advisors. Writing the paper was done collaboratively, I was the main supplier of content.

2. Chris Biemann, Kalina Bontcheva, Richard Eckart de Castilho, Iryna Gurevych, and **Seid Muhie Yimam**. Collaborative Web-Based Tools for Multi-layer Text Annotation. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, pages 229–256. Dordrecht, 2017. preprint: https://www.inf.uni-hamburg.de/en/inst/

ab/lt/publications/2017-biemannetal-hola-webbasedtools-preprint.pdf

- I co-author this publication, which contains an extended version of **Yimam** et al. (2013). In addition, I have contributed to the related work section, which compares a wide range of annotation tools. I do not have a contribution to the description of the "GATE Teamware" tool.

3. Richard Eckart de Castilho, Chris Biemann, Iryna Gurevych, and **Seid Muhie Yimam**. WebAnno: a flexible, web-based annotation tool for CLARIN. In *CLARIN Annual Conference*, Soesterberg, The Netherlands, 2014. Online: https://www.clarin.eu/sites/default/files/cac2014_submission_6_0.pdf

   - I co-author this publication, which contains an extended version of **Yimam** et al. (2014).

4. **Seid Muhie Yimam**, Heiner Ulrich, Tatiana von Landesberger, Marcel Rosenbach, Michaela Regneri, Alexander Panchenko, Franziska Lehmann, Uli Fahrer, Chris Biemann, and Kathrin Ballweg. New/s/leak – Information Extraction and Visualization for Investigative Data Journalists. In *Proceedings of ACL-2016 System Demonstrations*, pages 163–168, Berlin, Germany, 2016c. Online: http://www.aclweb.org/anthology/P16-4028

   - In this work my contributions are: 1) investigation and integration of different NLP components, 2) dataset preparation and integration, and 3) main coordinator to write the paper. The other authors are responsible to the frontend and API development of the tool. I have demonstrated New/s/leak at the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016) system demonstration track that was held in Berlin, Germany.

5. Darina Benikova, Uli Fahrer, Alexander Gabriel, Manuel Kaufmann, **Seid Muhie Yimam**, Tatiana von Landesberger, and Chris Biemann. Network of the Day: Aggregating and Visualizing Entity Networks from Online Sources. In *Proceedings of NLP4CMC at KONVENS2014*, pages 48–52, Hildesheim, Germany, 2014b. Online: https://hildok.bsz-bw.de/files/277/01_07.pdf

- In this publication I have participated in the development and integration of the different NLP components of the tool.

6. Kathrin Ballweg, Florian Zouhar, Patrick Wilhelmi-Dworski, Tatiana von Landesberger, Uli Fahrer, Alexander Panchenko, **Seid Muhie Yimam**, Chris Biemann, Michaela Regneri, and Heiner Ulrich. New/s/leak – A Tool for Visual Exploration of Large Text Document Collections in the Journalistic Domain. In *VIS conference 2016 collocated with the Workshop on Visualisation in Practice*, Baltimore, MD, USA, 2016. Online: http://www.aclweb.org/anthology/P16-4028

    - I co-author this publication, which contains an extended version of **Yimam** et al. (2016c).

7. **Seid Muhie Yimam** and Chris Biemann. Demonstrating PAR4SEM – A Semantic Writing Aid with Adaptive Paraphrasing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 48–53, Brussels, Belgium, 2018. Online: http://aclweb.org/anthology/D18-2009

    - This work is entirely carried out by myself. I have demonstrated Par4Sem at the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018) system demonstration track that was held in Brussels, Belgium. The other co-author in the publication is my supervisor.

1.4.2 Resources for Adaptive Technologies

In the process of exploring adaptive approaches, we collect different resources for the semantic writing aid tool, namely the complex word identification (CWI) and paraphrase ranking datasets. We collect the datasets to build an initial model for the application. Furthermore, we use the CWI datasets for the Complex Word Identification Shared Task 2018 that is organized as part of the BEA workshop co-located with NAACL-HLT'2018. We present the details of the data collections and experimental results in Chapter 6 and Chapter 7. The following are the list of publications regarding the CWI and paraphrase resources.

1. **Seid Muhie Yimam**, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. A Report on the Complex Word Identification Shared Task 2018. In *Proceedings of The 13th Workshop on Innovative Use of NLP for Building Educational Applications, NAACL 2018 Workshops*, pages 66–78, New Orleans, LA, USA, 2018. Online: http://aclweb.org/anthology/W18-0507

    - In this work my contributions are: 1) read and summarize papers of participating teams in the shared task, 2) main coordinator to write the paper, 3) build the baseline system, and 4) dataset preparation and publication.

2. **Seid Muhie Yimam**, Héctor Martínez Alonso, Martin Riedl, and Chris Biemann. Learning Paraphrasing for Multiword Expressions. In *Proceedings of the 12th Workshop on Multiword Expressions*, pages 1–10, Berlin, Germany, 2016b. Online: http://www.aclweb.org/anthology/W16-1801

    - This work is entirely carried out by myself. I have orally presented the paper at the 12th Workshop on Multiword Expressions (MWE 2016), co-located with ACL 2016 that was held in Berlin, Germany. The other co-authors in the publication acted as advisors.

3. **Seid Muhie Yimam**, Sanja Štajner, Martin Riedl, and Chris Biemann. CWIG3G2 - Complex Word Identification Task across Three Text Genres and Two User Groups. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 401–407, Taipei, Taiwan, 2017c. Online: http://www.aclweb.org/anthology/I17-2068

    - This work is entirely carried out by myself. I have orally presented the paper at the 8th International Joint Conference on Natural Language Processing (IJCNLP 2017) that was held in Taipei, Taiwan. The other co-authors in the publication acted as advisors.

4. **Seid Muhie Yimam**, Sanja Štajner, Martin Riedl, and Chris Biemann. Multilingual and Cross-Lingual Complex Word Identification. In *Proceedings of the International Confer-*

ence Recent Advances in Natural Language Processing, RANLP 2017, pages 813–822, Varna, Bulgaria, 2017b. Online: https://doi.org/10.26615/978-954-452-049-6_104

- This work is entirely carried out by myself. I have orally presented the paper at the 2017 International Conference on Recent Advances in Natural Language Processing (RANLP 2017) that was held in Varna, Bulgaria. The other co-authors in the publication acted as advisors.

1.4.3 Annotation Automation and Personalized NLP Applications

For rapid annotation, in addition to enhancing the user interface of annotation tools, we also integrate an automation component inside the annotation tool that can produce suggestions.

Furthermore, we investigate the utility of integrating a personalized and adaptive machine learning model into an NLP application where the training data are exclusively obtained from usage data. The following are the list of publications concerning the experiments of the annotation automation and personalized NLP application.

1. **Seid Muhie Yimam**, Richard Eckart de Castilho, Iryna Gurevych, and Chris Biemann. Automatic Annotation Suggestions and Custom Annotation Layers in WebAnno. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. System Demonstrations*, pages 91–96, Baltimore, MD, USA, 2014. Online: http://www.aclweb.org/anthology/P14-5016

   - This work is entirely carried out by myself and describes an extension based on my previous work (**Yimam** et al., 2013). I have demonstrated WebAnno at the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014) system demonstration track that was held in Baltimore, MD, USA. The other authors in the publication acted as supervisors and technical advisors.

2. **Seid Muhie Yimam**, Chris Biemann, Ljiljana Majnarić, Šefket Šabanović, and Andreas Holzinger. An adaptive annotation approach for biomedical entity and relation recognition. *Brain Informatics*, 3(3):157–168, 2016a. Online: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4999566/

- The technical part of this work is entirely carried out by myself, which is another extension based on my previous work (**Yimam** et al., 2015). The other co-authors acted as advisers and annotators (doctor-in-loop). I authored the technical description and the experiment and result sections; the description of the medical issues related to the application domain was written by others.

3. **Seid Muhie Yimam**, Chris Biemann, Ljiljana Majnarić, Šefket Šabanović, and Andreas Holzinger. Interactive and Iterative Annotation for Biomedical Entity Recognition. In Yike Guo, Karl Friston, Faisal Aldo, Sean Hill, and Hanchuan Peng, editors, *Brain Informatics and Health*, pages 347–357, London, UK, 2015. ISBN 978-3-319-23344-4. Online: https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2015-yimametal-bih-london.pdf

   - This work is entirely carried out by myself. I have orally presented the paper at the International Conference on Brain Informatics and Health (BIH 2015) that was held in London, UK. The other co-authors acted as advisers and annotators (doctor-in-loop).

4. **Seid Muhie Yimam**. Narrowing the Loop: Integration of Resources and Linguistic Dataset Development with Interactive Machine Learning. In *Proceedings of HLT-NAACL: Student Research Workshop*, pages 88–95, Denver, CO, USA, 2015. Online: http://www.aclweb.org/anthology/N15-2012

5. **Seid Muhie Yimam** and Chris Biemann. Par4Sim – Adaptive Paraphrasing for Text Simplification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 331–342, Santa Fe, NM, USA, 2018. Online: http://aclweb.org/anthology/C18-1028

   - This work is entirely carried out by myself. I have presented the paper in the poster session at the 27th International Conference on Computational Linguistics (COLING 2018) that was held in Santa Fe, New-Mexico, USA. The other co-author is my supervisor.

1.5 Organization of the Thesis

In Chapter 2, we review the foundations of annotation tools and best practices that should be adopted while developing the tools. We also discuss the building blocks of rapid annotation, adaptive annotation, and personalized NLP application research directions.

Part II of the thesis focuses on the techniques, approaches, and design principles of annotation tools, visualization tools, and personalized NLP applications. Chapter 3 presents how we develop WebAnno, a rapid annotation tool, which can easily be installed by NLP engineers and can be easily accessed from browsers by annotators without a need to install additional plugins. Furthermore, we describe the rapid annotation functionalities such as keyboard-based annotation, forward annotation, and an automatic annotation adjudication. The integration of automatic annotation suggestion in WebAnno is also explained in detail. In Chapter 4, we present the fundamental approaches of developing a personalized NLP application using an embedded machine learning model. Chapter 5 presents the different InfoVis tools that are used to collect training datasets during information exploration.

In **Part III**, we present the development of two different resources that are particularly important for the development of semantic-aware writing aid NLP application. Chapter 6 describes how we collect paraphrase resources using the Amazon Mechanical Turk crowdsourcing and evaluate the impact of the dataset for context-aware text paraphrasing. The dataset is later used in Chapter 9 to build a baseline system for an adaptive semantic writing aid tool. In Chapter 7, we describe how to collect and evaluate the complex word identification dataset, which is a prerequisite for the text simplification task. We use the dataset to build a baseline system for the adaptive text simplification use-case, mainly to automatically identify complex words or phrases.

Part IV of the thesis focuses on the experimental results and implications of rapid annotation and personalized NLP application approaches. In Chapter 8, we present the experimental results of integrating an adaptive machine learning model into annotation tools, reporting the main outcomes of the approach. In Chapter 9, We report the experimental results of employing an adaptive machine learning model into a personalized NLP application, namely a semantic text writing aid tool applied for the text simplification task.

Part V concludes the thesis by presenting the main contributions of the thesis and pointing

to the future outlook. Chapter 10 concludes about the tools, resources, experimental setups, and evaluation results obtained. It also briefly indicates possible future directions, mainly on how an adaptive learning approach can be utilized by broader NLP applications.

Study the past if you would define the future.

Confucius

2

Annotation and Automation Strategies

In this chapter, we concisely discuss the approaches and foundations of rapid annotation, adaptive annotation, and adaptive NLP applications. The building blocks of annotation tools, the different automation strategies for adaptive annotation, and the requirements of adaptive and personalized NLP applications will be described. We also illustrate the important concepts related to annotation and automation strategies that are going to be referred to throughout the remaining chapters.

2.1 Science of Annotations

As we have presented in Chapter 1, natural language processing (NLP) applications incorporate machine learning components, which in turn depend on labeled or annotated data.

Annotation in general and in particular corpus annotation is the process of adding extra information, or description (see Figure 2.1.1), to a raw text for several purposes (Hovy and Lavid,

2013). An annotation can be attaching a part-of-speech tag for each token in a text, labeling each named entity in a text, linking mentions or items that belong to the same category in a text (coreference annotation), to mention a few (Biemann et al., 2017). For NLP applications, annotations are mainly used to train a machine learning model, which then enables learning and predicting of annotations (labels) for unseen data. In the remainder of this thesis, the terms *annotation*, *label*, and *tag* are used interchangeably, which refer to an attachment of extra information to parts of a corpus (for example to words, phrases, sentences, and documents).

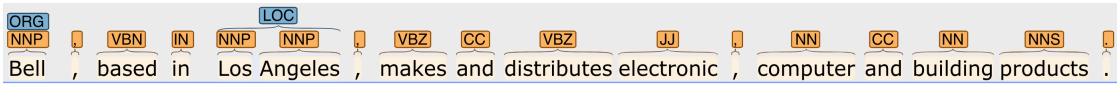


Figure 2.1.1: Examples of different annotations, for example **Bell** is annotated as **ORG** (organization for named entity annotation type) and as **NNP** (proper noun singular for POS tag annotation type). The visualization is based on the Brat annotation interface (Stenetorp et al., 2012), which is integrated as the frontend component of WebAnno (**Yimam** et al., 2013).

2.2 The MATTER Annotation Development Cycle

Annotation is not by any means a one-time task. It is rather an iterative and continuous process, which consists of problem definition, development of the annotation tool, collecting the actual annotation, evaluating the quality of the collected data, and repeating the whole process until the dataset meets the quantitative and qualitative requirements. The work of Pustejovsky and Stubbs (2012) describes this annotation process as the MATTER annotation cycle, shortened for *Model*, *Annotate*, *Train*, *Test*, *Evaluate*, and *Revise*.

The **model** phase is mainly concerned about the problem description that one has to specify before starting the annotation process. Like with any other software system, the requirements, as well as the expected inputs and outputs, should be first identified. By far, this is the most important aspect of the annotation development cycle. During the **annotate** phase, a specific set of attributes is identified that are going to be annotated to encode a structural description and the properties of the data. In the **train** and **test** phases, an algorithm is trained, and its performance is tested using the specified feature sets, respectively. During **evaluation**, results

are evaluated quantitatively. Finally, the whole cycle is revised based on the achieved results to further enhance the prediction performance of the model (**revision**).

2.3 Annotation Types

When adding annotations to text, the annotation value, the type of the annotation, and the region or location of the annotation in the original text should be identified unambiguously.

In this regard, the annotations can be broadly grouped into a span, relation, or chain type, particularly for standoff markup annotation. In **Standoff** annotations, the annotations are stored separately from the document, usually referred back to the text using offsets or positions, while **inline** annotation encodes the annotation directly inside the document's flow (Pianta and Bentivogli, 2004). In **span annotation type**, we need to know the offsets (begin and end position) of the span (parts of the text to be labeled).

In **relation annotation types**, two span annotations (or two portions of the text) are going to be linked together with a given relation value. If more than two span annotations are going to be connected to each other to show the relations among all the span annotations, it is a **chain annotation type**. For image and speech annotation, the target of the annotation is usually some region of the image or the sound wave.

2.4 Annotation Representations

The goal of annotation in the scope of language technology is to provide the tagged documents for a machine learning tool that learns to predict annotations or labels of the same kind to unseen data. In this respect, it is crucial to have annotations represented in a convenient format (representation) to be read easily by the NLP applications that incorporate a machine learning model. While there are some guidelines for annotation representations, not all NLP applications consume the same representations. Most of the time, every NLP application has their representation that is different from the formats the annotation tools are producing. For annotation tools, it is essential to support the following properties when representing the annotations.

1. **Human readable**: Annotations should be represented in a format that is easily readable by a human. If the formats are not easily readable, it is challenging to access the labels

and extract the required features for the target machine learning algorithm.

2. **Connection to the original text**: It should be easy to link the annotations back to the original text, which means that all the information such as the offset in the text, file encoding, and other relevant information should be included beside the annotations.

3. **Unambiguous representation**: If we have different annotation types attached to the same document, the annotation representation should make clear which annotation belongs to which annotation types.

4. **Storage and exchange**: It should be easy to convert either to other representations or to the target machine learning application.

Below are some of the annotation representations widely adopted in most of the linguistic annotation projects. The tab separated value (TSV) annotation format, a representation implemented in WebAnno, will be discussed separately in Chapter 3.

1. **The Graph Annotation Format (GrAF)**:GrAF is an instantiation of the linguistic annotation framework (LAF) paradigm (Ide and Suderman, 2014). The GrAF annotation format encodes annotations in a graph representation, which aims 1) to show the generality of the graph model for representing linguistic annotations, 2) to demonstrate how the graph-based model helps in merging and analyzing annotations, and 3) to propose it as an underlying model for linguistic annotations (Ide and Suderman, 2007). GrAF in particular and LAF in general, are better stated as abstractive and general conceptualizations of annotation formats but they are not easy to be adopted by existing tools as well as pose greater effort to understand by end-users.

2. **Text Encoding Initiative (TEI)**: TEI[1] is a consortium, which aims in the production of guidelines or standards for encoding texts in a machine-readable format. It is a very extensive encoding format having a strong ground in the humanities, social sciences, and linguistics. TEI provides elements, attributes, and other mechanisms for encoding different types for linguistic corpora.

[1] http://www.tei-c.org/index.xml

3. **The CoNLL-like formats**: The CoNLL file formats are mainly developed alongside a series of CoNLL shared tasks (Nivre et al., 2007) that have been taken place since 1999. In the CoNLL-like file Formats (e.g., CoNLL-U and CoNLL-X), annotations are encoded in multiple columns with three line types. The word lines represent the word (token), the annotation, and further information, which are separated by a single tab character. Blank lines mark the separation of sentences, while comment lines that start with the hash (#) character are used to add additional comments for each or block of annotation word lines.

4. **Format for Linguistic Annotation (FoLiA)**: FoLiA (van Gompel and Reynaert, 2013) is an XML-based format developed as part of the Dutch CLARIN project, having in mind that XML technologies such as *XPath*, *XSLT*, and *XQuery* can handle FoLiA annotations and make use of the hierarchical capabilities of annotations in XML (van Gompel and Reynaert, 2013). FoLiA supports four categories of annotations: 1) **Structure annotation** (inline annotation): mainly for document representations such as tokens, headers, paragraphs, sentences, and utterance. 2) **Token annotation** (inline annotation): which is an extension of structural elements applied, for example, to a single token that supports annotations such as part-of-speech tags, lemmatization, and sense annotation. 3) **Span annotation** (stand-off annotation): applied to a span of multiple tokens for annotations such as named entities, semantic roles, and chunking. 4) **Higher-Order Annotation** : Represent annotations on another annotation.

5. **The D-SPIN Text CorpusFormat (TCF)**: Both FoLiA (van Gompel and Reynaert, 2013) and TCF (Hinrichs et al., 2010) are products developed as a contribution to the European Common Language Resources and Technology Infrastructure (CLARIN) project. TCF is developed in the Deutsche Sprachressourcen-Infrastruktur (D-SPIN) project that focuses on representation formats to support interoperability between different web services in a corpus processing pipeline.

2.5 Annotation Frameworks

In the process of collecting training dataset, annotation tools or frameworks plays an important role. The annotation tool influences the speed of the annotation process and the quality of the labeled data. An annotation tool should be user-friendly, allows to complete annotations quickly, partly supports validating the annotation quality, and provides possibilities to import and export annotated documents. In the following sub-sections, we will discuss some of the properties and types of annotation tools in general.

2.5.1 Specific Versus General Annotation Tools

Most of the annotation problems are specific, where the guidelines, tag-sets and tags, text or corpus types are confined to a particular annotation problem. Consider annotating part-of-speech (POS) for English, or dependency parsing annotations for German. For such annotation problems, it is possible to quickly build a specific annotation tool with a list of appropriate tags and an annotation guideline. The annotation tools designed will have all the requirements and attributes demanded by the annotators (linguists). Such annotation tools are convenient for annotation problems that are very specific, limited to a specific natural language problem.

However, such specific annotation tools cause the following shortcomings:

1) Most of the requirements are restricted to the annotation problem specified, and it is challenging to customize to another annotation problem, which has similar aspects but differs slightly for some of the annotation needs. For example, if the annotation tool is build to support named entity annotation from news articles, it might not be possible to use it to annotate medical entities, which have extensive attributes.

2) Even for the same annotation problem (for example POS tag annotation), the tool fails to support the annotation of POS tag for a different language due to differences in grammar, tag-set, Unicode representation, and writing direction.

2.5.2 Collaborative and Distributive Annotation Strategies

Annotations can be undertaken either in a collaborative or distributive approach. In the **collaborative annotation** approach, two or more users are assigned to work on the same document

(Bontcheva et al., 2010; Cunningham et al., 2002, 2011; Ma et al., 2002). In a collaborative approach, there is no particular way of identifying which annotation is created by which user. While it is possible to set up user management, mainly for administrative purpose, it is not used to attribute a document to a specific user. The following annotation tools belong to such annotation framework.

1. **The Annotation Graph Toolkit (AGTK)** (Ma et al., 2002): AGTK allows annotation of a given file by a group of users, who can collaborate on an annotation project.

2. **The OLLIE application for collaborative annotation** (Tablan et al., 2003): The OLLIE client-server application supports collaborative corpus annotation, which relies on the GATE (Cunningham et al., 2002) annotation framework.

3. **GATE Teamware** (Cunningham et al., 2011): GATE Teamware is an open-source, web-based collaborative annotation tool that provided physically distributed annotators (with different user roles) the possibility to carry out complex corpus annotation.

4. **BRAT** (Stenetorp et al., 2012): BRAT is a web-based NLP-assisted annotation tool, where its client-server architecture allows a real-time collaboration annotation. Annotators can work on a single document simultaneously, visualizing each-others annotation in a real-time.

In the case of **distributed annotation** approaches, user management and document assignments are a crucial component of the annotation tool. There could be different ways of assigning a user to documents. If the annotators are reliable and precise in annotating the corpus, a document can be assigned to a single annotator. On the other hand, if there are multiple annotators where it is not clear if a single annotator is reliable or not (for example the annotators are not expert in the area, or there is no direct control of the annotators like in a crowdsourcing platform), it is better to distribute one document to multiple annotators. Distributing one document for multiple annotators helps to collect several annotations and decide on the final annotation based on a majority vote. In this case, the final annotation will be determined based on the annotation agreements. There are different measurements to compute inter annotator agreements for annotations. Popular inter annotator agreement metrics are **Cohen's Kappa**

(Cohen, 1960), **Fleiss' Kappa** (Fleiss, 1971), and **Krippendorff's Alpha (nominal)** (Krippendorff, 2011) (see Chapter 3 Section 3.2.5).

2.5.3 Standalone and Web-based Annotation Tools

Annotation tools can be broadly divided, based on the visualization support and installation requirements, into **standalone** (desktop) and **web-based** tools. Standalone annotation tools are intended to run on a predefined environment, for example on a specific operating system or relying on a given programming language pre-installation.

However, web-based annotation tools are designated to be installed in a server environment and will be accessed through a web browser to complete the annotation task. Web-based annotation tools do not require annotators to install the tool in their local machine. The fact that web-based annotation tools are accessed from a browser further enables to collect a large number of annotations.

Web-based annotation tools (both collaborative and distributive) are preferred for reasons such as 1) users will not need to install the tool in their machine, 2) users can access the tool from a different location over the Internet, and 3) it allows sharing the same document to multiple users.

2.5.4 Annotation tools for Crowdsourcing Platforms

The emergence of crowdsourcing introduces different strategies in designing annotation tools for crowd-workers. The design of an annotation tool for crowdsourcing is different from lab-based annotation tools in several aspects such as:

1) **Annotator control**: There is no direct control over the crowd-workers to conduct annotator training. The communication channel consists only of the annotation guideline and automatic or manual feedback on the annotation quality.

2) **Availability**: Annotation processes using crowdsourcing requires the annotation tool to be accessible all the time. For example, in the case of Amazon Mechanical Turk (MTurk), if the tool is not accessible especially once the task is started, it leaves the workers in confusion, they can neither complete the task nor get payment for the task they have completed so far.

3) **Annotation quality**: It needs close monitoring on the annotation process (for example

developing different qualification requirements, building a control dataset[2], blocking scammers, and so on) to ensure the quality of the annotated data.

4) **Domain experts**: Unlike with dedicated or lab-based annotation tools, the annotation process over crowdsourcing cannot reliably leverage domain expert.

There are usually two possibilities of annotation methods in crowdsourcing setups, at least in the case of the popular MTurk platform. One option is to embed the annotation tool as it is into the crowdsourcing platform. In this case, one can design the annotation user interface inside the crowdsourcing platform and separately upload the data (the documents for the annotation, annotation examples, and instructions or guidelines). A technical challenge of such an approach is that there is no easy and direct control over the user's interaction with the annotation tool during the annotation process.

The second option is to run the annotation tool on an external server and embed the tool as an HTML frame into the crowdsourcing platform. In this case, there is full control over the annotation process. The limitation of this approach is that the external server should remain available all the time until the annotation process is completed.

2.6 Rapid Annotation and Automation

The main objective of developing annotation tools is to make the annotation process easy and to produce quality annotation datasets. Recently, the need to have a rapid annotation tool is ever increasing due to the versatile nature of machine learning-based applications and fast prototyping (Aziz et al., 2008; Biemann, 2005; Kaufmann and Bernstein, 2007; Stenetorp et al., 2012). We can characterize rapid annotation as follows:

1. **Automatic suggestions**: Instead of awaiting the user to mark parts of a text and add the required label, rapid annotation supports the user by automatically identifying the annotation targets (parts of the text to be annotated) and providing a suggestion for labeling. Such automatic suggestion reduces the time required to look for the expected portion of the text and adding the correct label. Furthermore, if the suggested labels are wrong, the

[2]**Control dataset** is an annotation example that are used to control the quality of the annotation, where the answer to the task is determined in advance.

annotator is expected only to change the labels instead of marking the positions, at least. However, if the position detected is wrong, the annotator can easily reject the suggestion.

2. **Support collaborations**: If the annotation project is large, a lot of users are going to participate in the annotation process. The annotation tool should support 1) in identifying agreements between different annotators, 2) in automatically resolving conflicts, and 3) in providing the progress of the annotation.

3. **Early termination**: Annotation tools should support the possibility of providing the most representative texts during the early stage of the annotation process. Selecting such examples earlier helps to terminate the annotation process before annotating the whole corpus. This approach is known as **active learning** (Settles, 2010), and annotation tools with active learning capability promote a rapid annotation paradigm.

4. **Browser-based annotation**: Annotation tools designed as a desktop application have a lot of shortcomings. If users have an option to access the annotation tool from a browser, hence from different devices including mobiles, the annotation process will be completed much faster as it is not required to install separately for each annotator. Web-based applications, in general, are also easy to use.

We can conclude that rapid annotation is mainly expressed by an automatic suggestion and correction capability as well as active learning strategies. Below, we will discuss the two approaches that are used to generate suggestions in annotation tools.

2.6.1 Resource-Based Rapid Annotation

One possible option to generate suggestions in an annotation tool is to use external resources such as dictionaries or lexicons, ontologies, and gazetteers. For example, to annotate medical documents such diseases, symptoms, causes, medicines, or protein instances, one can use domain specific dictionaries to uncover medical terms and provide suggestions automatically. When such resources are not available, annotation tools could also offer recommendations based on histories, i.e., from previous annotations (building resources on the way).

2.6.2 Machine Learning for Rapid Annotation

When annotation tools integrate a machine learning model to generate suggestions, the annotation process can be completed much faster than it could be done with a tradition annotation process. There are different methods of incorporating machine learning components inside annotation tools.

1. **Using existing datasets**: In some NLP domains, there might exist training datasets to build an initial model. The main purpose in this direction is either to increase the volume of the dataset or to improve the quality of dataset (Andriluka et al., 2018). In this situation, it is easy to build a model based on the existing datasets and produce suggestions for annotators to correct them. Once enough training examples are further collected, it is possible to retrain or update the embedded model to increase its accuracy.

2. **Based on usage dataset**: When there is no training dataset to build an initial model, the best approach is to start with an empty model but make use of the usage data collected during the annotation process to train a new model. This approach is one of our focus in this thesis that we extensively explore and discuss in the remaining chapters.

2.7 Adaptive NLP Applications

In the scope of this thesis, an adaptive natural language processing application is an application that continuously improves its internal machine-learning model from usage data.

Let us briefly describe the difference between an adaptive NLP application and adaptive annotation tools. In the case of the adaptive NLP application, there is no separate tool to collect the training dataset, but the training dataset is derived from how people perform a task that is supported by the adaptive application. In adaptive annotation tools, the objective is to help the annotator in collecting the required training data by providing suggestions. The annotation tool is adaptive because it learns from the existing annotation to provide suggestions. The user can either accept or correct suggestions, which in turn are used to update the model. However, the task of annotating is not an end by itself, but rather an artificial task for annotators.

In this work, we will illustrate an adaptive NLP application in Chapter 9 by constructing a semantic writing aid tool. Writing aid tools are natural candidates for adaptive NLP because 1)

writing is vital to produce text electronically, 2) existing writing aid tools also provide technical support in the form of suggestions from thesauri, and 3) users already accept such type of suggestions in writing tasks.

However, the existing writing aids do not support updating the underlying resources by utilizing the information collected from the user but for the most basic functionalities, such as adding words to spelling dictionaries manually. There is also a need for extended support of lexical semantics: In the area of writing aid tools, there are no systems that offer context-sensitive reformulation assistance similarly to the widespread spelling and grammar checkers. Existing lexical dictionaries and thesauri in word processors are not sufficient since 1) they do not consider the context and 2) they are generally limited to single (input) words. We will explore how to build semantic writing aid tools that integrate an adaptive model to support document composing. We target two tasks, namely complex word or phrase identification and text simplification to demonstrate the adaptability of NLP applications using semantic writing aid tools.

Part II

From Rapid Annotation Tools to Adaptive and Personalized NLP applications

To carry out a rapid annotation process and to develop personalized and adaptive NLP applications, building an appropriate tool is vital. In Part II, we present different annotation tools and adaptive NLP applications we have developed during the thesis work.

In Chapter 3, the design and development of a rapid annotation tool called **WebAnno**, which is a web-based, general, and distributed annotation tool will be presented. WebAnno also incorporates an automation component specifically designed to address adaptivity during the annotation process.

In Chapter 4, the **Par4Sem** tool, which is specifically developed for an adaptive semantic writing aid NLP application will be discussed.

Finally, in Chapter 5, some information visualization (**InfoVis**) tools that are used to collect training datasets beyond their primary purpose of data exploration will be described.

*Well I took a look at U-Compare and found the display gave me an error when I loaded a large file. I started looking at the code to solve the problem only to discover the source is only partially open... In looking around for a UIMA / Brat integration I have found the fully open source **WebAnno** project.*

James Kitching, HivE Mind'S EYE Open Source Project

3

Development of Rapid Annotation Tools for Dataset Collection

The development of an appropriate tool, which should be easy to use and allows a rapid annotation process, is essential for the collection of a dataset. In this chapter, we discuss the design approaches we have followed in developing **WebAnno**: a web-based, distributed, and generic annotation tool. Then, we present the automation component that is mainly designed to support the adaptive annotation. We also briefly compare WebAnno with existing annotation tools based on its annotation functionalities as well as its automation support.

WebAnno forms the technological basis for many of the user-facing experiments described in Chapters 8. Apart from being a required prerequisite for the research problems in this work, the WebAnno software is a contribution of its own, being widely used in the NLP community.

3.1 WebAnno - A web-based and Distributed annotation tool

WebAnno[1] (Eckart de Castilho et al., 2014, 2016; **Yimam** et al., 2013, 2014) is an annotation tool that we have developed in collaboration with the UKP group at TU Darmstadt. It is a generic annotation tool with the following main functionalities: 1) It has a built-in annotation support for the basic linguistic annotation types such as part-of-speech (POS) tagging, dependency layer, lemmata, named entities, and coreference annotation, 2) It supports the creation and annotation of any generic non-linguistic annotation types (example Human Phenotype Ontology Annotation[2], Relation Emotion Annotation for Fiction[3], and Spatial Analysis of Literary Texts[4]), 3) It supports multiple file formats for importing and exporting annotated documents, 4) Besides the annotation interface, it has correction, automation, and curation components, 5) It has a user-friendly interface for project management and annotation progress monitoring, and 6) It is an open-source tool accompanied with a detailed documentation that includes user-guidelines and developer support. The live demo of WebAnno demonstrates its functionality with example projects[5].

3.2 Architecture of WebAnno

WebAnno is a web-based, generic, and distributed annotation tool that supports the rapid annotation paradigm. It is **web-based**, in a sense that all the annotation and project management related activities are performed within a web browser (Chrome and Safari are officially supported). Once WebAnno is installed, the creation of annotation projects, annotation layers, users as well as the actual annotation task can be completed from a web browser. Users can access the tool over the internet, without the need to install WebAnno on local machines or without installing any browser plugins. It is a **distributed** annotation tool as it is possible to distribute annotation documents to multiple users who can work on the same document without modifying annotations of each other. Moreover, it is a **generic** annotation tool, which can

[1] https://webanno.github.io/
[2] https://webanno.github.io/webanno/use-case-gallery/hpo-crowd-annotation/
[3] https://webanno.github.io/webanno/use-case-gallery/reman/
[4] https://webanno.github.io/webanno/use-case-gallery/spatial-annotation/
[5] username/password: guest/guest http://ltdemos.informatik.uni-hamburg.de/webanno/

be used to work on a large variety of linguistic as well as non-linguistic annotation tasks.

3.2.1 Backend components

The backend component of WebAnno is responsible mainly 1) to store, retrieve, and update annotation documents, 2) for user management, and 3) for annotation layer management. Annotation documents are stored in a file system while the rest of the information such as user and layer information is stored in a database. WebAnno is implemented using the Java programming language, which includes mainly the apache UIMA[6], Spring[7], apache Wicket[8], and Hibernate[9] frameworks. Figure 3.2.1 displays the main components and processes of WebAnno.

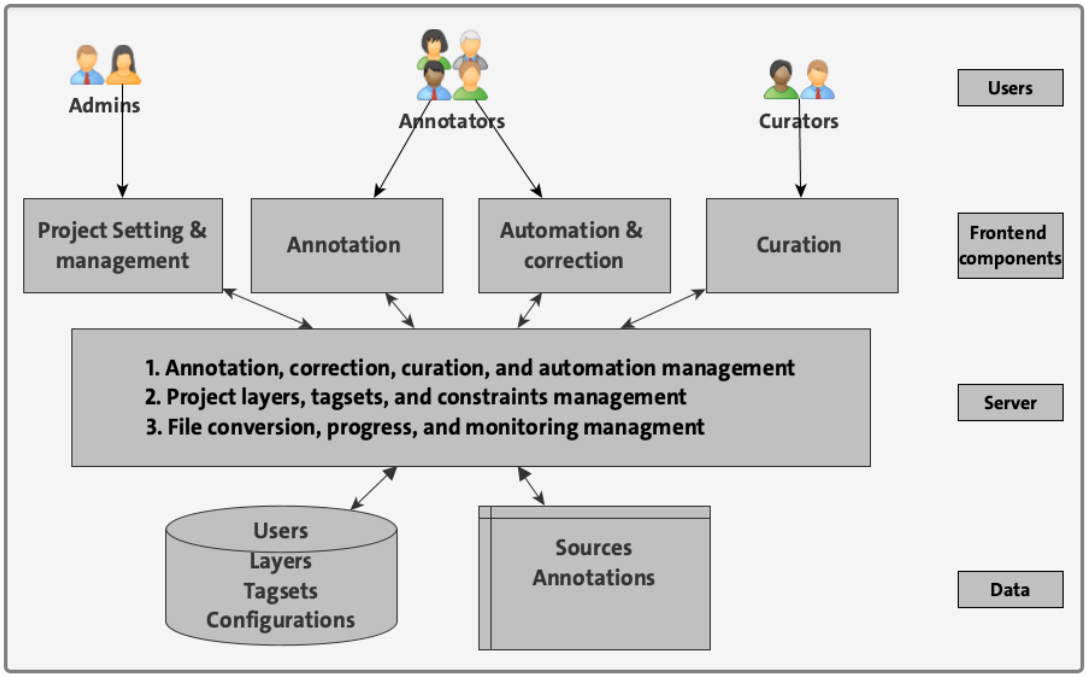


Figure 3.2.1: System architecture of WebAnno: organized in user, front end, back end, and persistent data storage management.

[6] https://uima.apache.org/
[7] https://spring.io/
[8] https://wicket.apache.org/
[9] http://hibernate.org/

3.2.2 User and Project Management

WebAnno includes different type of users with different roles and responsibilities. **Administrative** (super) users are those responsible for creating and managing projects, managing other users, and monitoring project progress. **Project admin** users are those who are specifically assigned to handle a given project that includes importing documents into the project, assigning annotator users, creating annotation layers and tag-sets, and controlling and monitoring project progress. **Annotator** users primarily perform the annotation task on a project only for the documents they are assigned to work. Finally, **curators** are special users who have to adjudicate annotation documents from different users. It is also possible to assign multiple roles for a single user, for example assigning a user both as a project admin and a curator.

3.2.3 Annotation Interface

The front end of WebAnno presents an interface where users interact with the document to annotate, which is based on the Brat annotation visualization component (Stenetorp et al., 2012). The Brat interface presents the documents split into different sentences, displaying the sentences in separate lines. It provides lists of annotation elements, which are going to be attached to the current annotation document (see Figure 3.2.3). The interface further allows exporting the annotated document, navigating to different documents and projects, and configure different settings such as how many sentences to display at a time and set the size of the annotation editor (see Figure 3.2.2). Below, we discuss the different types of annotation elements (an annotation layer with a span, relation, and chain types).

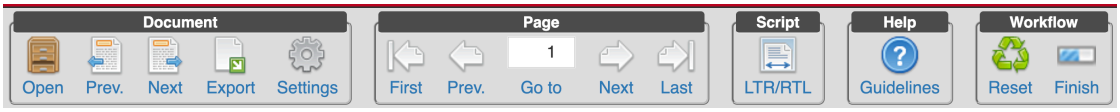


Figure 3.2.2: The WebAnno menu bar showing the different operations allowed in the annotation interface (such as exporting annotations, navigating documents and sentences, and editing the different annotation settings).

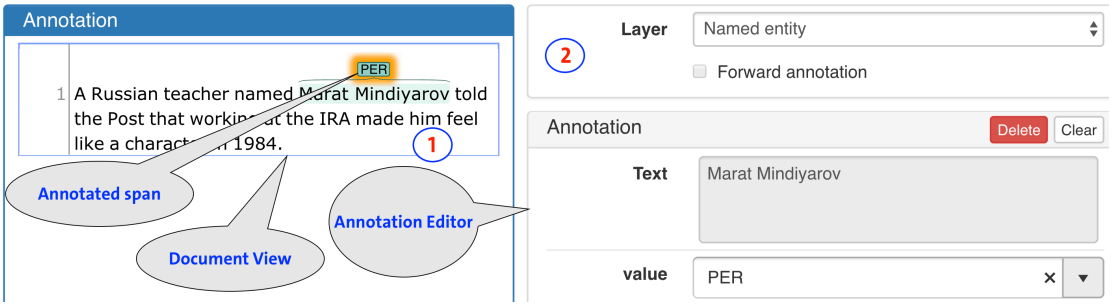


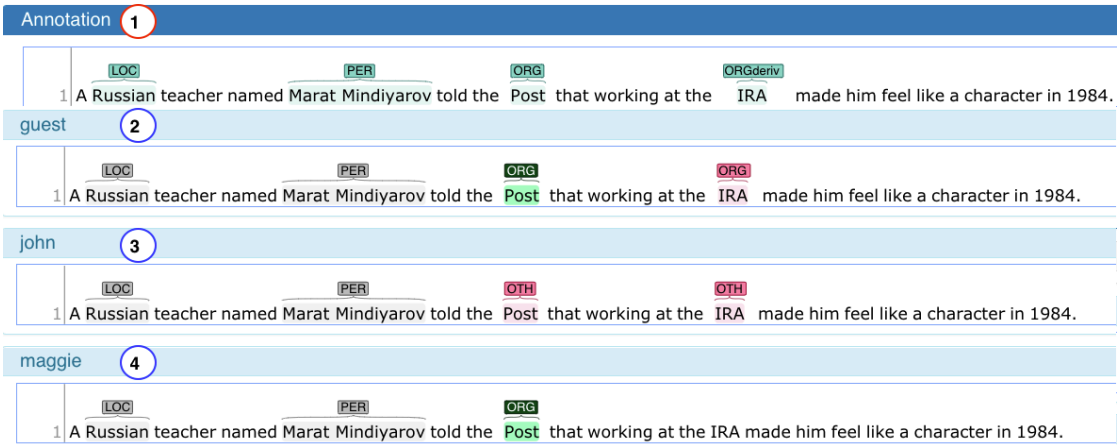
Figure 3.2.3: The Annotation page of WebAnno displaying the Document view (1) and the Annotation editor (2).

3.2.4 Curation Interface

Like the annotation interface, the curation interface allows curators to inspect annotations collected from different users. The curation component tries to adjudicate annotations from multiple users automatically, and the user interface presents a visually informative analysis of curation results, which are color-coded (for example red for disagreement and green for agreement, as it can be seen from Figure 3.2.4) based on the adjudication results. In Figure 3.2.4, the lower section shows the annotations from three users (the user **guest** –(2), the user **john**– (3), and the user **maggie** –(4)) the upper annotation shows the final curated results((1)) . The curator can also add new annotations when other annotators miss them at all. The output of the curation is supposed to be the final quality dataset to be exported and used by NLP applications to build a machine learning model.

3.2.5 Project Monitoring and Agreement Analysis

The project monitoring page provides top-level project progress and the assignment of projects and documents to different users (see Figure 3.2.5). The agreement management interface presents an informative analysis of the annotation status concerning the inter-annotator agreements among different users (see Figure 3.2.6). The agreement is computed pair-wise among all annotators and for all documents. Hence, for each document, annotations are inspected based on the positions or offsets and the actual labels assigned by the annotators. For **span** annotation types, agreement between two annotators are determined when the positions (begin and end



is

Figure 3.2.4: The Curation page of WebAnno displaying annotations from three users and the final curated annotation.

offset) and the labels assigned are the same. As we can see from Figure 3.2.6, (1) shows the focus of the agreement analysis on the specific annotation type and attribute (here **Named Entity – value**). And (2) shows the different options of inter-coder or inter-rater reliability measures supported in WebAnno (**Cohen's Kappa** (Cohen, 1960), **Fleiss' Kappa** (Fleiss, 1971), and **Krippendorff's Alpha (nominal)** (Krippendorff, 2011)). The actual agreement measures are shown at the upper part of the table ((3)) while different statistics such as the number of annotation values provided by the annotator and the total number of annotation values covered by the two annotators are displayed at the bottom of the table ((4)). If the two annotators are not assigned the same documents (hence no common annotation values) or if a given annotator does not annotate at all the provided documents, the table further presents extra pieces of information as it is shown in (5) and (6) respectively.

3.3 Annotation Layers

In WebAnno, annotations are performed using annotation layers. An annotation layer corresponds to one of the annotation types, namely span, relation, or chain types. Figure 3.3.1 shows the annotation layer settings editor of WebAnno.

The **span annotation** types are those, which are used to mark parts of texts (example token,

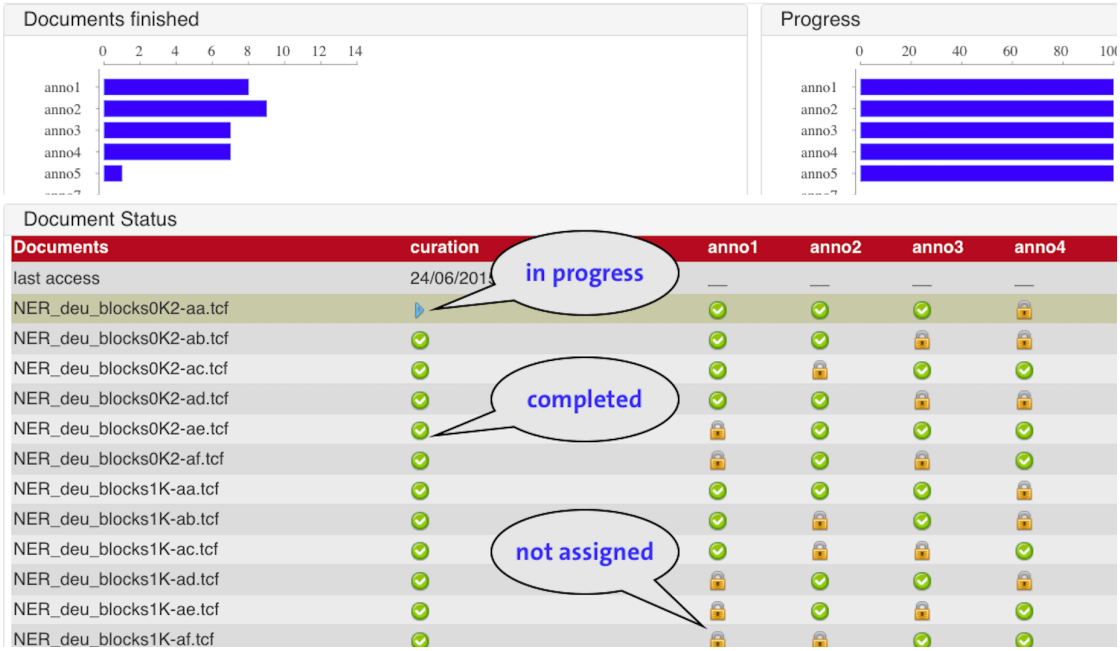


Figure 3.2.5: The Monitoring page of WebAnno displaying the progress of annotation documents for each user.

sub-tokens, multiple tokens or even a sentence) with a begin and end offset. The **relation annotation** types are used to mark links between two span annotations with a directed arc. When more than two span annotation types are going to be connected to indicate a relation between them, it is defined as a **chain annotation** type. In general, WebAnno provides span, arc, chain, and semantic annotation layers. These three generic annotation types allow producing multiple properties on annotations, e.g., supporting rich morphological annotations and creating semantic annotations.

3.3.1 Span Annotation

When the span annotation covers tokens (lexical words in a text, usually separated by a space), the annotation layer is defined as token-based annotation layer, such as lemma (John agree / John agrees) and POS tag (NNP VBZ / John agrees) annotations. When the annotation covers multiple tokens, it is a multi-token annotation layer, for example named entity annotation layer (PER / Ms. Haag ORG / Rolls-Royce Motor). An-

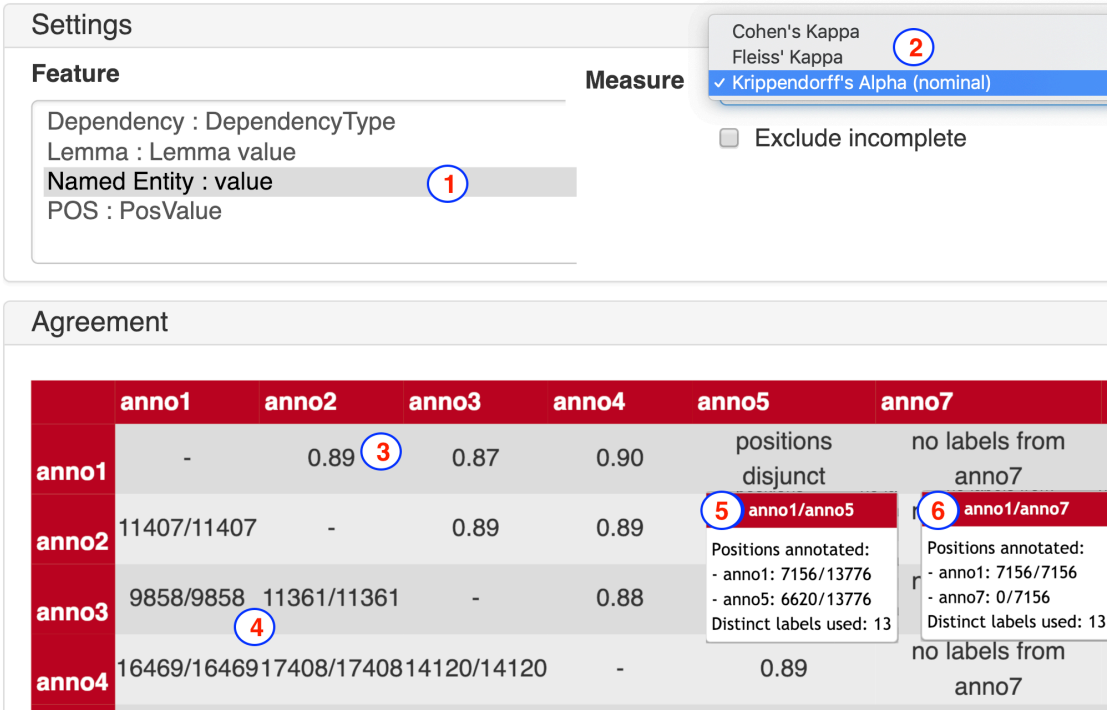


Figure 3.2.6: The Agreement view of WebAnno displaying the annotation agreement among different users.

notation layers can also be defined as sub-token annotation types, when it is required to annotate sub-parts of a token, such as morphological annotation layers like *prefixes* and *suffixes* (pre root suf unspeakable). Annotation layers can also cover multiple sentences, for example sentiment analysis (negative | giz2000 | 429 | Tue Jun 02 03:32:45 UTC 2009 lley my dentist is great but she's expensive[10]) annotation[11]. Zero-width annotations are used to insert arbitrary markers at a specific position of the text. Span annotation layer can also be defined as **stacked** annotation to allow attaching multiple annotations on the same span (same begin and end offset) of a text, example ambiguous value for a named entity (told the OTH ORG Post that).

[10]Here, the sentence is annotated with 4 different attributes of sentiment layer, i.e., opinion (negative), user id (giz2000), tweeter Id (429), and date (Tue Jun 02 ...) separated by the | character.

[11]http://help.sentiment140.com/for-students/

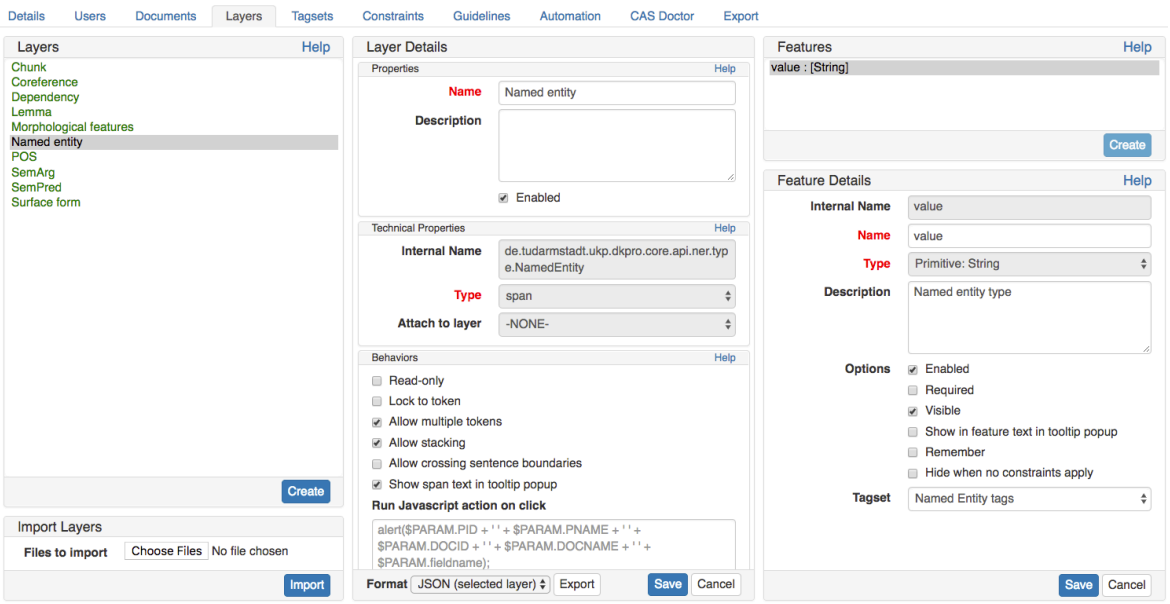
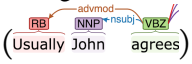


Figure 3.3.1: WebAnno annotation layer setting panel.

3.3.2 Relation Annotation

To label relation annotation, WebAnno requires first to create the span annotation types. For example, we know that dependency annotations are drawn based on POS tags (see Figure 3.3.2). So, the relation annotation layer depends on the existing span annotation layers (

). Relation annotation can be easily created by drawing a line from the source span annotation (the governor) to the target span annotation (the dependent).

WebAnno also allows reversing the direction of the arc. Furthermore, hovering or pointing the mouse cursor over the relation annotation shows the yield of the relation while hovering over the span annotation shows the yield of the POS tag annotation. For example, in Figure 3.3.2, pointing on the relation annotation *SB* shows the yield of relation between the *NNPS* and *VBD* POS annotations and pointing on *VB* for the token *remain* shows the yield of the POS tag annotation. The **yield of** annotation is used to provide information about the tokens transitively covered by the outgoing relations. This is useful mainly to present all the governed tokens that the head of a particular dependency relation dominates.

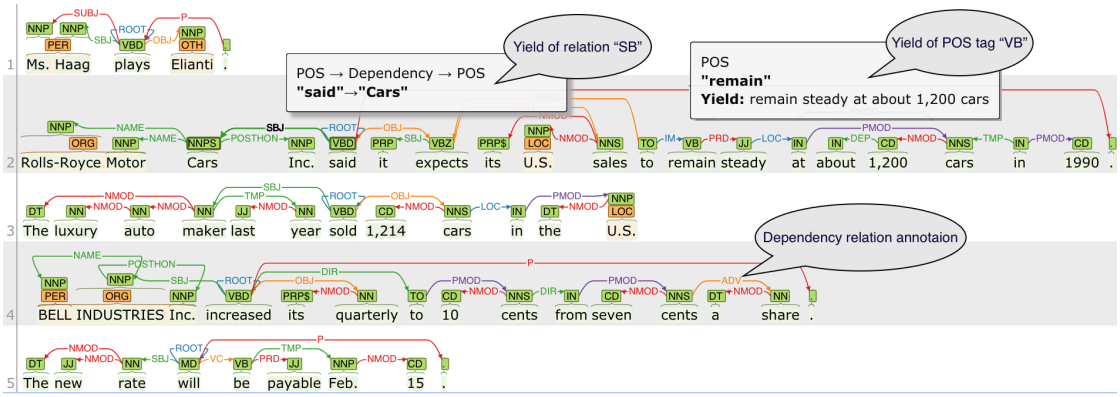


Figure 3.3.2: Dependency relation annotations over POS span annotations.

3.3.3 Chain Annotation

Chain annotations are similar to the relation annotation except that the number of spans connected to each other is not limited to two, as it can be seen from Figure 3.3.3 for the coreference annotations (van Deemter and Kibble, 1999). Every chain annotation is represented with its own color code. The chain annotation type is essential when we want to keep track of annotations that belong to the same category.

3.3.4 Semantic Annotation and Constraint-based Annotation

The span annotation type in WebAnno is further enriched to support semantic role labeling and event-related annotation tasks. Semantic annotation is enabled using slot attributes, which allow a span annotation (called a **slot owner** or **roles**) to link to multiple span annotations (called **slot fillers** or **arguments**) with specified argument values. Figure 3.3.4 shows the annotations as well as the annotation editor in WebAnno ((1)) compared to how it is visualized with the AllenNLP semantic role labeling[12] tool ((2)).

WebAnno supports annotations based on tag-sets (a collection of tags), which might comprise hundreds of tags (categories or labels to attach as annotations to a span type) based on the domain and need of the annotation problem. In this case, even if we have an auto-complete ability to display tags during annotation, it is much better to constrain or limit the tag-set based

[12] http://demo.allennlp.org/semantic-role-labeling

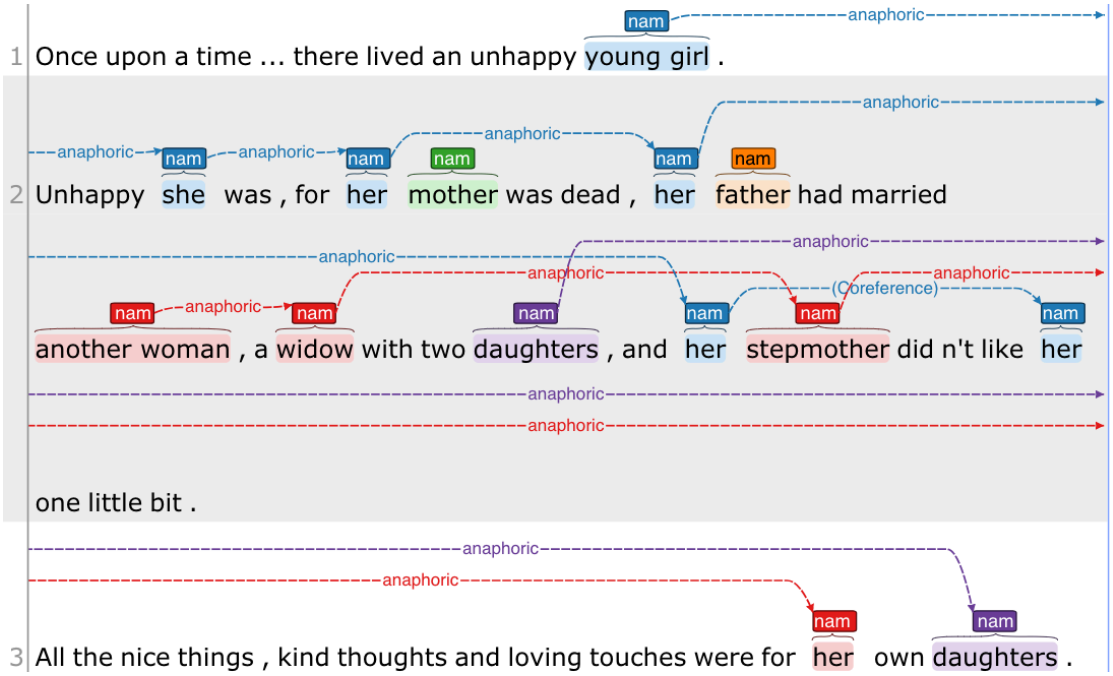


Figure 3.3.3: Coreference annotation as a chain annotation type.

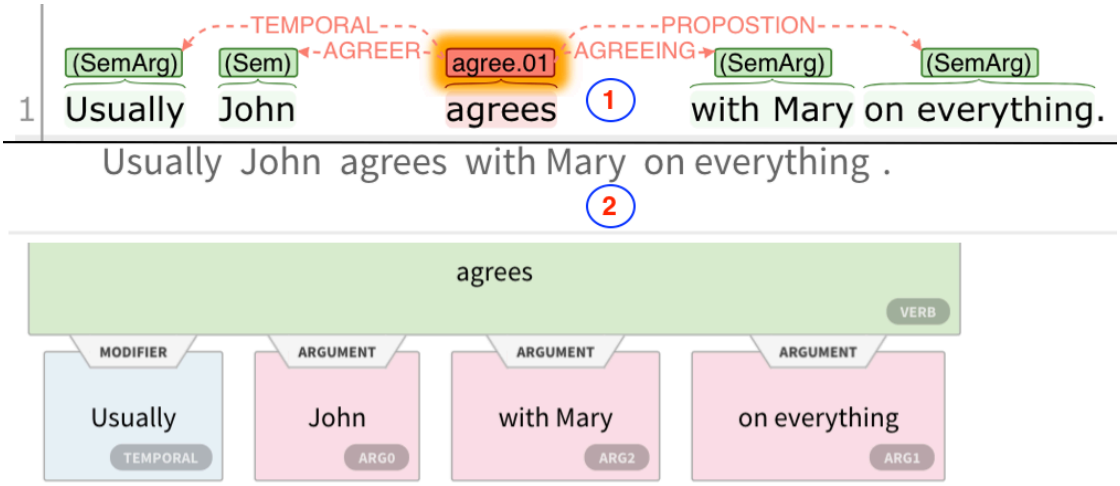


Figure 3.3.4: Semantic role labelling annotations in WebAnno vs. the AllenNLP Semantic Role Labeling tool.

on some pre-existing contexts. For example, when annotating dependency annotation, the relation values usually depend on the governor (source) and dependent (target) POS tags. Simi-

larly, the sense of a semantic predicate determines the available argument types. The constraint functionality helps to sort the tags based on the provided context where valid tags appear at the top of the list. Details on semantic annotation and constraint application in WebAnno are discussed in Eckart de Castilho et al. (2016).

3.4 Importing and Exporting in WebAnno

WebAnno supports different file formats or representations to import and export the documents along with the annotations. The representation ranges from simply allowing to import only the texts to more advanced ones that include the annotation and annotation layers. Currently, the tool supports formats such as Text, CoNLL, TCF, XML, TSV, and TEI, see Section 2.4 in Chapter 2.

Customized tab-separated values (TSV) is an annotation format specifically implemented for WebAnno. It allows importing and exporting almost all types of annotations, along with the annotation layers in a human-readable file format. More specifically, the TSV file format supports 1) importing and exporting the annotation types or layers at the header of the file, 2) allows to export token, sub-token, and multiple token annotations, all of them in a separate column, 3) enable to export and import relation annotation which includes the governor and dependent span annotation that are referenced by the line numbers, 4) support semantic annotations where each of the arguments and roles are represented in separate columns, and 5) allow exporting sentences separately from the annotations in their own lines.

The following are the main differences between CoNLL and TSV formats. 1) TSV include the annotation layers and their attributes at the header of the file. The headers help to understand the type of annotation expected in the file and to create the associated layers in WebAnno. The WebAnno TSV reader will check first if the expected annotation layers in the header are already defined inside WebAnno. 2) TSV supports sub-token annotations. 3) TSV is much more comprehensive than CoNLL format that includes span, relation, and chain annotation formats. It also includes representations of multiple annotations (on the same token and the same annotation layer).

As it can be seen from Figure 3.4.1, (1) shows the different annotations in the WebAnno annotation UI with annotations from different layers. When the annotations are exported with

the TSV file format, all the annotation types or layers are presented at the header ((2)) while all the annotation instances are presented in a separate column ((3)). The token and offset information are also presented ((4)).

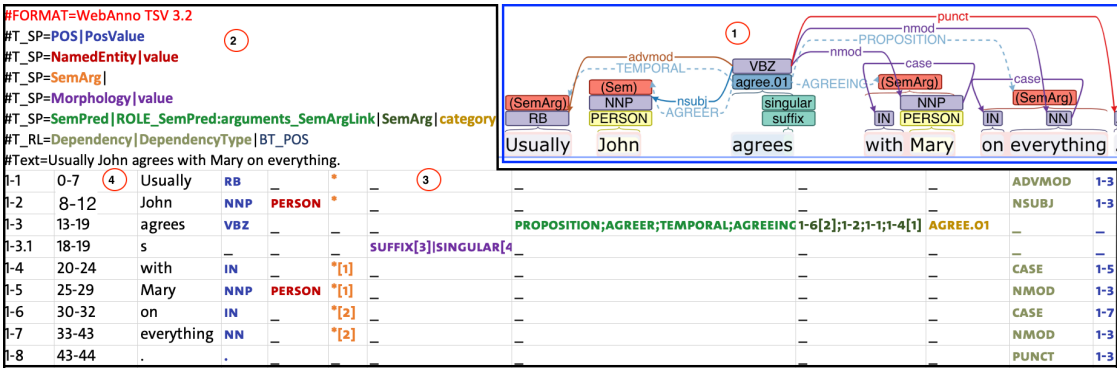


Figure 3.4.1: Annotation view and the exported TSV file. Headers include the different annotation layers. The sentence is given before each annotation. The first column displays the token ID, which can be used as a reference for other annotation, for example for dependency annotations. The second and third columns present the token offset and the token values. Other columns show the annotations, in the order of the layers at the header.

3.5 Rapid Annotation Support in WebAnno

The focus of rapid annotation is to conduct annotation tasks much faster than with conventional approaches, without affecting the quality of the datasets. Rapid annotation in WebAnno is materialized in several ways such as 1) designing a user interface that facilitates ease of use and supports minimal interaction during annotation, 2) supporting on-demand information access and automatically saving annotations without user intervention, 3) enhancing the auto-forward annotation approach, 4) supporting repeat-annotation criterion, and 5) integrates an annotation automation and correction component.

3.5.1 Cleaner User Interface of WebAnno

Unlike many annotation tools, WebAnno focuses on presenting fewer user interfaces at a time and limit the user interactions as minimal as possible. For example, in the Brat annotation tool,

one has to first select spans of texts and a pop-up dialog will be displayed to complete the annotation task. This process requires that the annotator has to move around the dialog (in case it covers the original text), perform the annotation task, and manually close the dialog. When this procedure is completed for every single annotation unit, it ends up taking too much time to complete the annotation task.

In WebAnno, all the required components for the annotation, correction, automation, and curation interfaces are presented automatically when it is required. For example, the annotation editor is visible as soon as the user highlights text spans or the tags for relation annotation types are displayed when the user draws a line between the source and the target span annotations. There is no component to manually open and close that requires more time for the annotation interaction.

3.5.2 On-Demand Information Access and Auto-save operations

Annotators usually will either quickly be introduced about the annotation process or an extensive training might be given to them depending on the type and domain of the annotation problem. WebAnno is designed with internal supports that users can get further information while annotating the documents. The project admin can upload annotation guidelines, which can be referred to anytime during the annotation process.

Furthermore, tooltips, a user interface component with a short description, are included in each smaller possible annotation components, to provide hints such as information about the given annotation layer or why two or more annotators disagree on a given annotation. Moreover, WebAnno does not have an explicit **Save** button to save or update the annotations, they are persisted immediately once the editing operation is completed.

3.5.3 Auto-forward Annotation

Most of the linguistic annotation tasks are performed on a single token and annotations are selected from a pool of tags in a tag-set. For example, while annotating part-of-speech tags, one might start from the first token in the sentence and continue annotating until the end of the sentence. Highlighting a single word and manually choosing the appropriate tag from the list is time-consuming. For such types of annotation tasks, WebAnno supports an **auto-forwarding**

and **key-binding** annotation support where each tag is associated with a short-cut of a key on a keyboard. The auto-forward functionality works when annotating token-based span annotations.

For example, four tags (of a POS tag-set) start with a letter "N" such as *NN*, *NNS*, *NNP*, and *NNPS*. Pressing the N key on a keyboard only one time will select automatically *NN* while pressing it three times will automatically annotate the text with the tag *NNP*.

Furthermore, it will automatically advance or move the annotation cursor to the next word (auto-forwarding), which eventually progresses until the end of the annotation document. The auto-forward functionality greatly reduces the annotation time at least by half of the total time it needs to complete without the auto-forward and key-binding functionalities.

3.5.4 Repeat Annotation

When annotating documents, for example, those containing technical terms that appear several times in the document as well as in the whole project, there is no need to separately annotate every occurrence of the terms or phrases. WebAnno introduces a **repeat annotation** functionality where such terms or phrases are automatically searched and annotated, but in the form of suggestions. We deliberately do not accept and include the suggestions or recommendations as a final annotation; instead, the annotators should inspect and approve them if they agree that the annotation is correct based on the annotation guideline. The approval method enables not to accept all forms of annotation suggestions indiscriminately based on the repeat-annotation method, as it might result in overly assuming that the other instances also receive the same annotation, which is not the case in some situations, for example, if the context of the terms or phrases is different. For example, the token **works** can be tagged as "VBZ" for "Verb, 3rd person singular present" in some occasion while it might also be tagged as "NNS" for "Noun, plural".

The repeat annotation is also extended to the relation annotations, where the relation annotation is automatically suggested if 1) the two span annotations co-occur in the same document or project and 2) the underlying texts where the span annotation co-occurs are the same.

3.5.5 Machine Learning Based Annotation Automation

The automatic annotation suggestion component is designed mainly to enhance the annotation efficiency while ensuring the quality of annotations. The fundamental design principle of our approach is a **split pane** (Figure 3.5.1) that displays automatic annotation suggestions in the **suggestion pane** (lower part) and only validated or manual annotations in the **annotation pane** (upper part). In this way, we force annotators to review each automatic suggestion to avoid overlooking wrong suggestions.

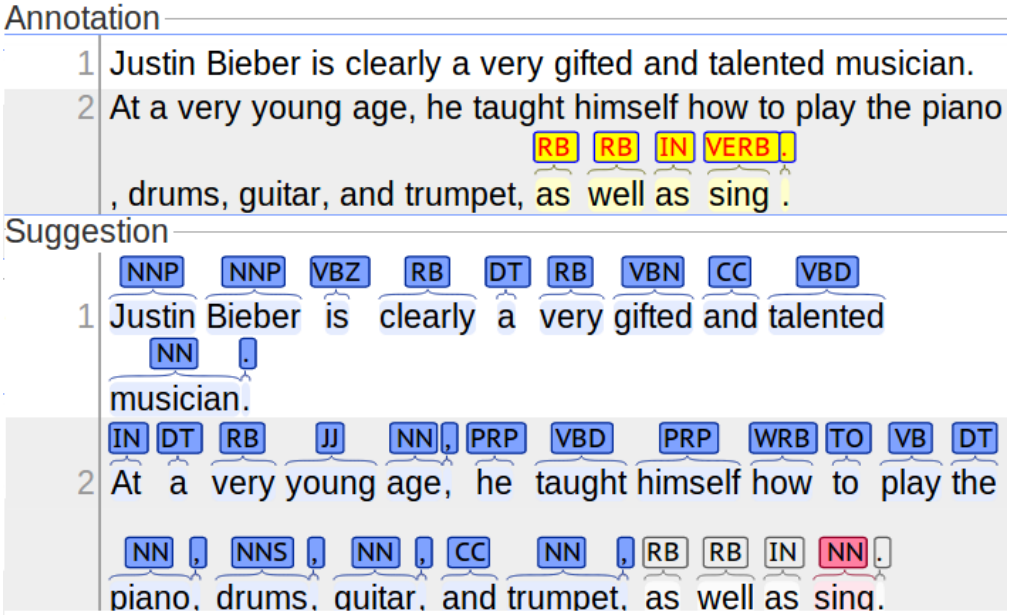


Figure 3.5.1: Split pane GUI for annotation automation. Upper: the Annotator pane for annotation, which should be completed by the annotator. Lower: the Automation pane displaying predictions or automatic suggestions, and coding their status in color. This example shows automatic suggestions with part-of-speech. Unattended annotations are rendered in blue, accepted annotations in grey and rejected annotations in red. Here, the last five POS annotations have been attended, four have been approved by clicking on the suggestion, and one was rejected by annotating it in the upper pane.

We distinguish three methods of automatic annotation suggestion.

Correction mode In the **correction** mode, it is possible to import documents annotated by arbitrary external tools (for example documents that are automatically annotated by a ma-

chine learning model) and present them to the user in the suggestion pane of the annotation page. This type of automation is specifically appropriate for annotation tasks where the pre-annotated document contains several possibilities of annotations, and the annotator's task is to select the correct annotation. Automation by correction allows to leverage specialized external automatic annotation. Thus, the tool is not limited to the integrated automation mechanism.

Repetition mode In **repetition** mode, if the annotator highlights a word or a phrase in the annotator panel, other occurrences of the same word or phrase are highlighted in the suggestion pane. To apply the annotation to the other occurrences of the same word or phrase, the user can just click on them in the suggestion pane. This basic suggestion method is realized using regular expressions or pattern matching.

Learning mode For the **learning** mode of automation, we integrate a machine learning annotator into the tool to display automatic suggestions in the suggestion pane. We have incorporated the MIRA machine learning algorithm (Crammer and Singer, 2003), an extension of the perceptron algorithm for online machine learning, which allows for automatic suggestions of span annotations. MIRA is selected because of its relatively lenient licensing, its excellent performance even on small amounts of data, and its capability of allowing incremental classifier updates. The architecture of WebAnno is flexible to integrate further machine learning tools besides MIRA.

3.5.6 Configuration of Machine Learning Based Automation in WebAnno

The workflow to set up an automatically supported annotation project consists of the following steps:

Configuring features. For the machine learning tool, it is required to define classification features to train a classifier model. We have designed a graphical user interface (GUI) where a range of standard classification features for sequence tagging can be configured. The features include morphological features (prefixes, suffixes, and capitalizations), N-grams, and also to consider other layers as a feature (for example POS annotation as a feature for named entity recognition). While these standard features do not lead to a state-of-the-art performance on specialized tasks, we have found them to perform very well, for example for the POS tagging,

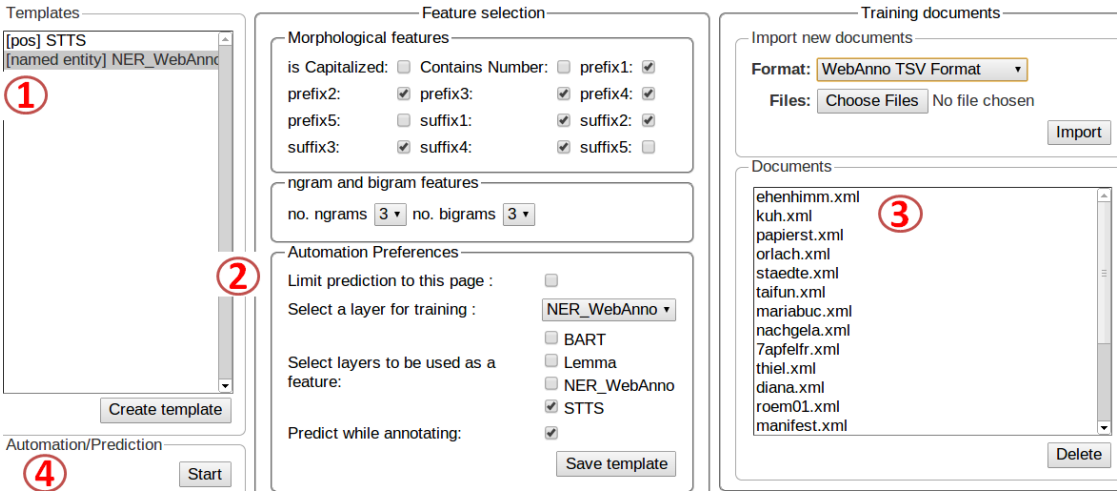


Figure 3.5.2: Configuring an automation project: 1) layers for automation 2) different features 3) training documents 4) start training classifier

named entity recognition, BioNLP entity recognition, and text chunking tasks. Figure 3.5.2 shows the feature configuration in the project settings. It is also possible to import **dictionary-like** collections that are going to be used to generate a binary feature.

Importing training documents. We offer two ways of providing training documents to train the embedded classifier model: importing an annotated document from external sources in one of the supported file formats, such as CoNLL, TCF, TSV, or XMI; or using internal (to WebAnno) annotation documents in the same project that already have been annotated and curated. In the case of internal annotation documents, it will be automatically added to the existing training documents once the curator marked them as **completed**.

Starting the automation process. Once features for a training layer are configured, and training documents are available (either from external sources or from internal annotation that are then marked as completed by the curator), the automatic annotation process can be launched. The process can be started manually by the administrator from the automation settings page, and it will be automatically re-initiated when additional documents for training become available in the project. While the automatic annotation is running in the background, users still can work on the annotation task without being affected by the automation process. Training and creating a classifier will be repeated only when the feature configuration is changed or when a

new training document is available.

Displaying results on the monitoring page. After the training and automatic annotation process is completed, detailed information about the training data such as the number of documents (sentence, tokens), the features used for each layer, the F-score based on held-out data (which are automatically created during the training process), and the classification errors are displayed in the monitoring page. The user interface also displays information about the status of the training process as (**not started**, **running**, or **finished**). The information displayed helps to estimate if the automation process is helpful or not for the specified project.

3.6 Related Work

GATE Teamware Bontcheva et al. (2010) is an annotation tool that supports quality management, annotator management, and support of a large set of annotation layers and formats. It is mostly web-based, but the annotation is carried out with locally downloaded software. The GATE Teamware system is heavily targeted towards template-based information extraction. It sets a focus on the integration of automatic annotation components rather than on the interface for manual annotation.

General-purpose annotation tools like MMAX2 (Müller and Strube, 2006) or WordFreak (Morton and LaCivita, 2003) are not web-based and do not provide annotation project management. They are also not sufficiently flexible regarding different annotation layers. The same holds for specialized tools for single annotation layers, which we cannot list here for the sake of brevity.

Anafora by Chen and Styler (2013) is a recent web-based annotation tool for event-like structures. Specifically, it supports the annotation of spans and *n*-ary relations. Spans are anchored on the text while relations exist independently from the text and consist of slots that can be filled with spans. Annotations are visualized using a colored text background. Selecting a relation highlights the participating spans by placing boxes around them. Anafora is not suited for annotation tasks that require an alignment of the semantic structures with syntactic structures such as constituent or dependency parse trees.

Brat by Stenetorp et al. (2012) is a web-based annotation tool with a focus on collaborative annotation. The tool supports spans and *n*-ary relations (also called **events**). Annotations are

visualized as boxes and arcs above the text. Multiple annotators can simultaneously work on the same annotations instead of being isolated from each other. However, this removes the ability to calculate the inter-annotator agreement. All annotation actions are performed through a pop-up dialog, which necessitates many actions even for simple annotations. While in principle the support for semantic annotation in Brat through the *n*-ary relations is good, there is no support for guiding the user through rich semantic tag-sets, e.g., by showing only relevant tags based on the annotation's context. This problem is aggravated by an annotation dialog popping up for each action. The frontend of WebAnno integrates the Brat visualization interface.

If you want to prototype an application, compile regular expressions,
if you want to participate in a shared task, collect enough datasets,
if you want a real-world application, build an adaptive system.

from our experience...

4

Adaptive and Personalized NLP applications

Most of the natural language processing (NLP) applications such as question answering, information retrieval, machine reading, and text simplification include a machine learning model, where the datasets that are usually collected a priori using a separate and dedicated annotation tool are used to train the model. While such an approach is adequate for most of the cases, let us recall (from Chapter 1) some of the limitations of collecting datasets separately:

1. **Time consuming**: First of all, it is often required to develop a dedicated tool, provide training for annotators and continuously oversee the annotation process. Then, the data also should be verified incrementally, usually before the complete dataset is obtained. All of these activities are **time-consuming**.

2. **Expensive**: Developing annotation tools or customizing existing annotation tools to collect the dataset for a target NLP application is also expensive. The tool usually should be implemented primarily to annotate the dataset required for the target NLP application.

Then, based on the domain of the NLP application, annotators should be trained or at least an adequate annotation guideline should be prepared beforehand. Finally, the annotation process itself costs money, mainly paid for the annotators. We can see that this adds up to make the collection of datasets for the target NLP application **very expensive**.

3. **Concept or semantic drift**: Most of the time, the collected data are assumed to fit the NLP application at the current time, or even worse, based on analysis or studies done in the past. Naturally, the requirements of the application, as well as the nature of data required, changes over time. This means that the collected data at a particular time will gradually become obsolete or fails to produce correct predictions, which is known as **concept** or **semantic drift**.

Our instantiation of an adaptive NLP application is a text simplification interface. Specifically, we support editor functionalities for making texts lexically simpler by supporting users in paraphrasing texts, focussing on lexical replacements (as, e.g., opposed to syntactic transformations). The task of lexical simplification was chosen because of its straightforward definition and its variability between users and user groups, making it a good candidate for personalization.

First, we briefly describe the traditional annotation tool-based data collection process related to identifying complex words or phrases for text simplification. Then we explain our adaptive semantic writing aid tool, which generally is used to provide personalized and adaptive paraphrasing capability for text composing. As a use-case, we experiment with text simplification. For the text simplification experiment, we have integrated the tool into the Amazon Mechanical Turk (MTurk) crowdsourcing platform to obtain paid workers to conduct the text simplification experiment.

Note that, in this chapter, we mainly focus on the development of the user interface of the NLP applications and integrated tools. The collected datasets, experimental results, and analysis of results will be presented in Part III and Part IV of the thesis.

4.1 Complex Word Annotation Tool for Crowdsourcing

In lexical text simplification, the first task is to identify difficult or complex words or phrases that could pose challenges to understand the text for target readers. In one way, experienced people such as teachers or professionals (such as news writers from Newsela[1]) can discover the difficult part of a text and replace them with simpler words or phrases. On the other way, the target readers themselves can determine the complex words or phrases they did not understand and request simpler substitutes.

We have designed a complex word or phrase identification (CWI) tool that supports an easy identification of complex phrases in a text. The tool is explicitly intended to be integrated into the Amazon Mechanical Turk crowdsourcing platform, where crowd workers are requested to identify complex phrases.

4.1.1 Complex Phrase Identification Interface

For a lexical text simplification task, the most straightforward approach to manually identify the complex phrases is to use the mouse cursor and highlight parts of the text, i.e., complex phrases (CPs), that are supposed to cause difficulty in understanding the text. Figure 4.1.1 shows the user interface that is used to collect CPs from MTurk workers. The tool allows displaying five to ten sentences to the annotators, which is going to be completed as a single task called **human intelligence task - HIT**, so that it is possible to identify CPs based on their contexts. Users can easily highlight the difficult words and the selections they made are displayed in a separate text field, while the selections are highlighted in yellow background color in the main text, at the same time.

In addition to the CPs, workers are also requested to answer questions related to their native language and their level of English that is used for further experiments (see Chapter 6) related to complex word identification task.

The instructions are displayed at the bottom of each task (see Figure 4.1.2), which can be considered at any time during the annotation process. The instructions also include examples that demonstrate how the worker should deal with the annotation task.

[1] https://newsela.com/

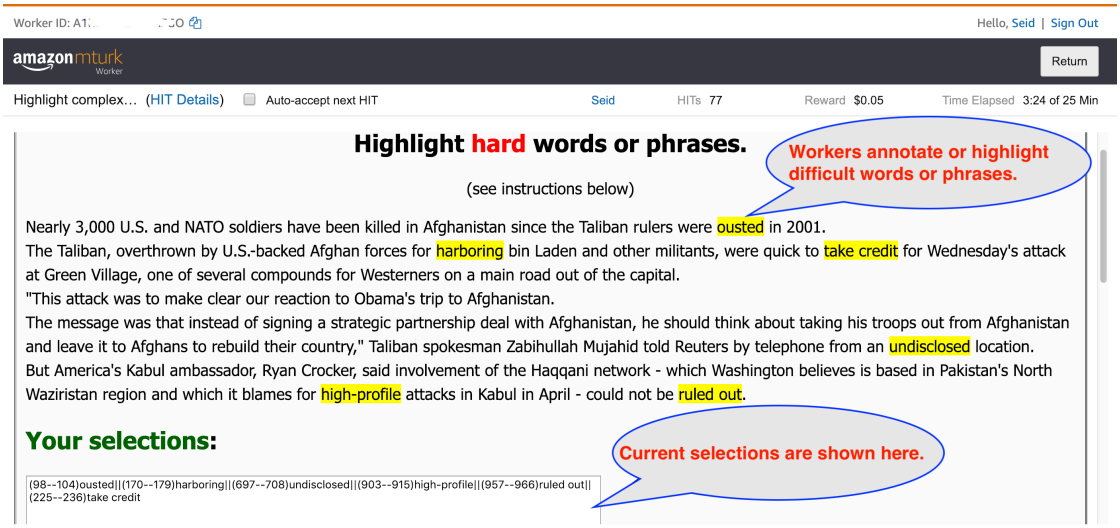


Figure 4.1.1: User interface for CP identification using MTurk: the actual task (the HIT).

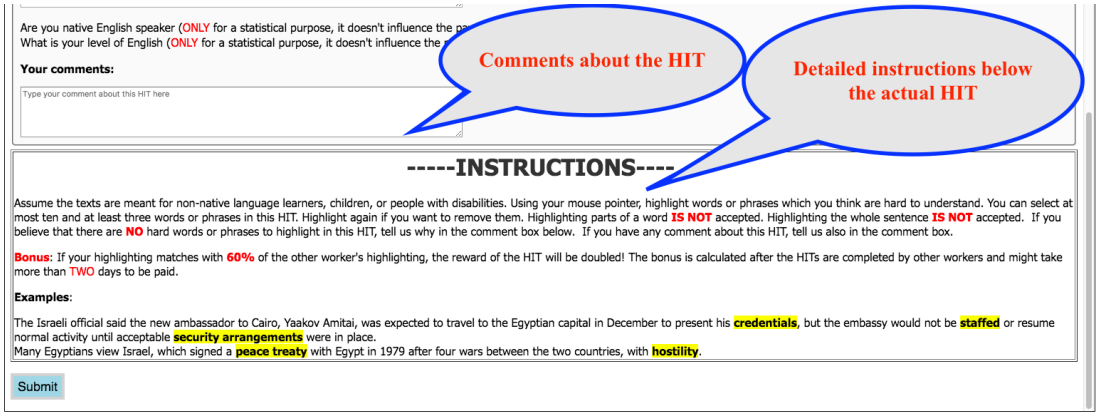


Figure 4.1.2: User interface for CP identification using MTurk: the instruction with examples.

4.2 Paraphrasing and Lexical Substitution Collection Tools

Most of the conventional paraphrasing and lexical simplification tools are based on a design where target words (complex phrases) are manually identified in advance and either 1) users are expected to type the possible alternative phrases by themselves or 2) candidates are presented for users to select the best matching simple candidate for the complex phrase. Figure 4.2.1 shows

a simple example where target words for paraphrasing are highlighted (**sit down** and **puzzle out** as shown in a) and the possible candidates are displayed in a list (see the candidates such as **work out**, **work,...** in b). We select 5 target units both single words and multi-word expressions or phrases specifically noun and verb phrases.

The user interface is designed as shown in Figure 4.2.2, where target units are highlighted in different colors and the possible candidates along the target units at the top are displayed. The user can easily select the appropriate candidate from the list or provide their own substitute if none of the offered suggestions can replace the target word without altering the original meaning of the text.

The tool is implemented as an HTML frame that can be embedded in the MTurk browser and it can support multiple languages as it is implemented with a variable-based placeholder to import the input data. Target units in the text are marked explicitly with an **HTML** tag for highlighting and list of candidates for each target words. In Chapter 6, we describe how candidates are provided, the statistic of the collected dataset, and some experimental results.

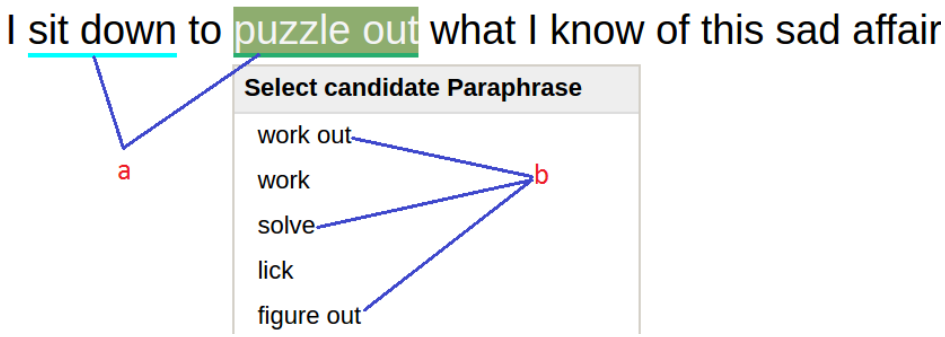


Figure 4.2.1: Targets for paraphrasing (a) and candidate paraphrases (b).

4.3 Adaptive Paraphrasing Tool for Semantic Writing Aid

Instead of collecting paraphrase datasets in a separate tool (cf. Section 4.2), we have designed **Par4Sem**, a semantic writing aid tool that can collect training dataset for paraphrasing from usage. Here, Par4Sem dealt mainly with the semantic annotation problem, using an adaptive, integrated, and personalized annotation process.

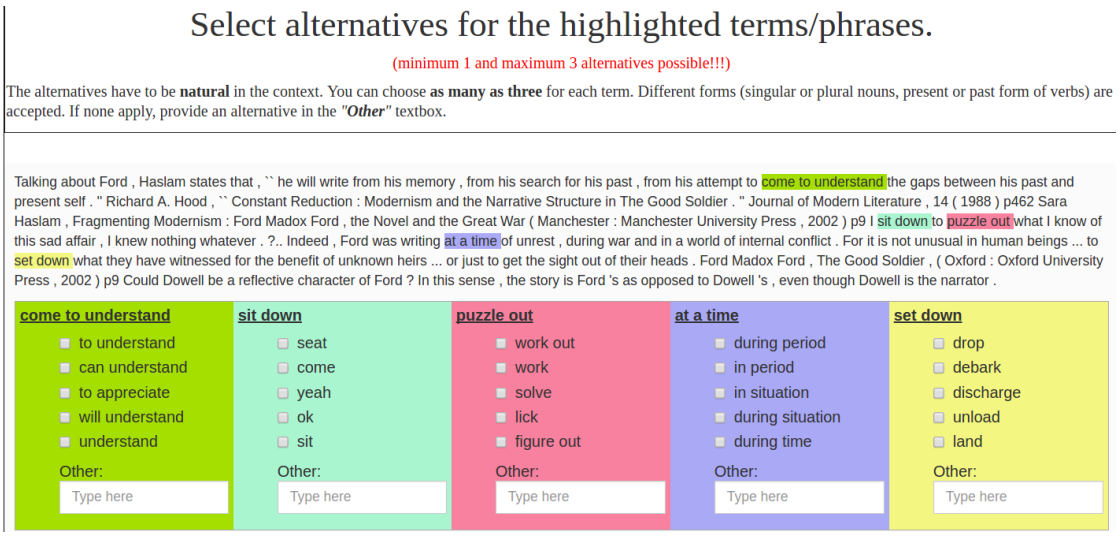


Figure 4.2.2: User-interface for context-sensitive paraphrase selection.

By **adaptive**, we mean that target applications, such as text simplification, do not require pre-existing training data, rather it depends on the usage data from the user. The machine learning model then adapts towards the actual goal of the application over time, entirely based on the feedback collected from the user interaction as training examples.

Instead of developing a standalone annotation tool, the collection of training examples is **integrated** into a real-world application. When the dataset collection is embedded into the target NLP application, related issues such developing annotation guidelines, developing an annotation tool, and controlling the annotation process will not be required.

Furthermore, our approach is **personalized** in the sense that the training examples being collected are directly related to the need of the user. It is true that the NLP application will not benefit from the adaptive machine learning component at the beginning. After all, the question is not: how good does the system perform *today*? It is rather: how good will it perform *tomorrow* after we use it today?

In general, the development of such adaptive approaches has the following benefits:

Suggestion and correction options: Since the model immediately starts learning from the usage data, it can begin to predict and suggest recommendations to the user immediately. Users can evaluate and correct suggestions that in turn help the model to learn from these corrections.

Less costly: As the collection of the training data is based on usage data, it does not need a separate annotation tool and the usual annotation process iteration.
Personalized: It exactly fits the need of the target application, based on the requirement of the user.
Model-life-long learning: As opposed to static models that will be deployed only once using the static training data, adaptive models incorporate more training data the longer they are used, which should lead to better performance over time.

Par4Sem is specifically designed as a semantic writing aid tool that relies on an adaptive paraphrasing component, which is used to provide context-aware lexical paraphrases during text composing. The tool includes specifically two adaptive models, namely target identification (aka *complex unit identification*) and candidates re-ranking (aka *paraphrase re-ranking*).

The adaptive target identification component is based on a classification algorithm, which learns how to automatically identify target units (such as words, phrases or multi-word expressions), that need to be paraphrased. When the user highlights target words, it is considered as a training example (*usage data*) for the adaptive model. The adaptive ranking model is based on a learning to rank machine learning algorithm, which is used to re-rank candidate suggestions provided for the target unit. We rely on existing paraphrase resources such as PPDB 2.0, WordNet, a distributional thesaurus, and word embeddings (see Section 4.4.1.1) to generate candidate suggestions.

4.4 System Architecture of Par4Sem

The **Par4Sem** system consists of a backend, frontend, and API components. The backend component is responsible for NLP related pre-processing, adaptive machine learning model generation, data storage, etc. The frontend component sends requests to the backend, highlights target units, presents candidate suggestions, and sends user interaction to the database. The API component transforms the frontend requests to the backend and returns responses to the frontend. Figure 4.4.1 shows the three main components of Par4Sem and their interactions.

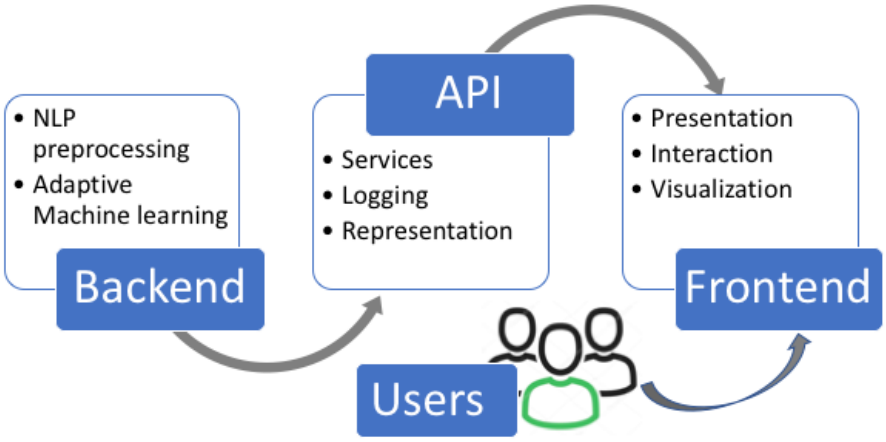


Figure 4.4.1: The main components of **Par4Sem**

4.4.1 The Backend Component

The backend component consists of several modules. For the adaptive paraphrasing system, the first component is to identify possible target units (such as single words, phrases, or multi-word expressions). For our lexical simplification use-case, the initial target unit identification machine learning model is trained with the datasets obtained from **Yimam** et al. (2017b,c, 2018) (cf. Chapter 7). The adaptive target identification unit then continues learning from the usage data (when the user highlights portions of the text to get candidate paraphrase suggestions).

Once the target units are marked or recognized (by the target unit identification system), the next step is to generate the possible candidate suggestions for the target unit (**candidate paraphrases**). The candidate suggestion module includes candidate generation and candidate ranking sub-modules. Section 4.4.1.1 discusses our approaches to generate and rank candidate paraphrases in detail.

4.4.1.1 Paraphrasing Resources

Paraphrase resources are datasets where target units are paired with a list of candidate units that are equivalent in meaning, possibly ranked by their similarity score. We refer to the work of Ho et al. (2014) about the details on how paraphrase resources are built, but we briefly discuss the different types of paraphrase resources that are employed to generate candidate suggestions for

Par4Sem.

PPDB 2.0[2]: The Paraphrase Database (PPDB) is a collection of over 100 million paraphrases that were automatically constructed using a bilingual pivoting method. Bilingual pivoting is described in Bannard and Callison-Burch (2005) as "look at what foreign language phrases the English translates to, find all occurrences of those foreign phrases, and then look back at what other English phrases they translate to". The recently released PPDB 2.0 includes improved paraphrase rankings, entailment relations, style information, and distributional similarity measures for each paraphrase rule (Pavlick et al., 2015).

WordNet[3]: We use WordNet synonyms, which are described as "words that denote the same concept and are interchangeable in many contexts" (Miller, 1995), to produce candidate suggestions for a given target unit.

Distributional Thesaurus – JoBimText[4]: We use JoBimText, an open source platform for large-scale distributional semantics based on graph representations (Biemann and Riedl, 2013), to extract candidate suggestions that are semantically similar to the target unit.

Phrase2Vec: We train a Phrase2Vec model, a variant of Word2Vec, (Mikolov et al., 2013) using English Wikipedia and the AQUAINT corpus of English news text (Graff, 2002). Mikolov et al. (2013) pointed out that it is possible to extend the word based embeddings (Word2Vec) model to phrase-based model using a data-driven approach where each phrase or multi-word expression is considered as an individual token during the training process. We have used a total of 79,349 multiword expressions[5] and phrase resources as given in **Yimam** et al. (2016b). We train the Phrase2Vec embeddings with 200 dimensions using skip-gram training and a window size of 5. We have retrieved the top 10 words that are similar to the target units as candidate suggestions.

4.4.1.2 Adaptive Machine Learning

Par4Sem comprises two adaptive machine learning models. The first machine learning model is used to identify target units (**target adaption**) in the text while the second machine learning

[2] http://paraphrase.org/#/download

[3] https://wordnet.princeton.edu/download

[4] http://jobimtext.org

[5] https://www.inf.uni-hamburg.de/en/inst/ab/lt/resources/data/learn-multi-word-paraphrasing.html

model is used to rank candidate suggestions (**ranking adaption**). Both models utilize usage data as a training dataset.

The target adaption model predicts target units based on the usage data (training examples) and sends them to the frontend component, which is then highlighted for the user. If the user replaces the highlighted target units, they are considered as positive training examples for the next iteration.

The ranking adaption model first generates candidate paraphrases using the paraphrase resource datasets (see Section 4.4.1.1). As all the candidates produced from the paraphrase resources might not be relevant to the target unit in the context, or as the number of candidates to be displayed might be unreasonably large (for example the PPDB 2.0 resource alone might produce hundreds of candidates for a single target unit), we re-rank the candidate suggestions using a learning to rank adaptive machine learning model. Figure 4.4.2 displays the process of the adaptive models while Figure 4.4.3 displays the pipeline (as a loop) that is used in the generations of the adaptive models.

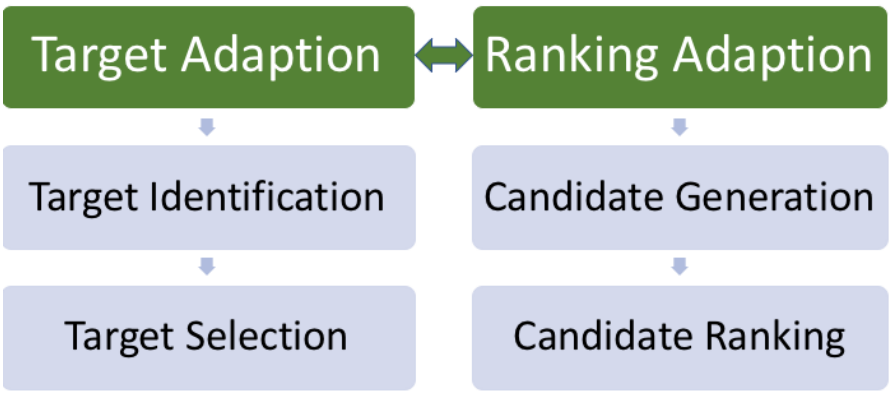


Figure 4.4.2: The main and sub-processes of target and ranking adaption components of **Par4Sem**.

The whole process is iterative, interactive, and adaptive in the sense that the underlying models (both target adaption and ranking adaption) get usage data continuously from the user. The models are updated for each iteration, where n examples are collected in a **batch mode** without a model update, and provide better suggestions (as target units or candidate suggestions) for the next iteration.

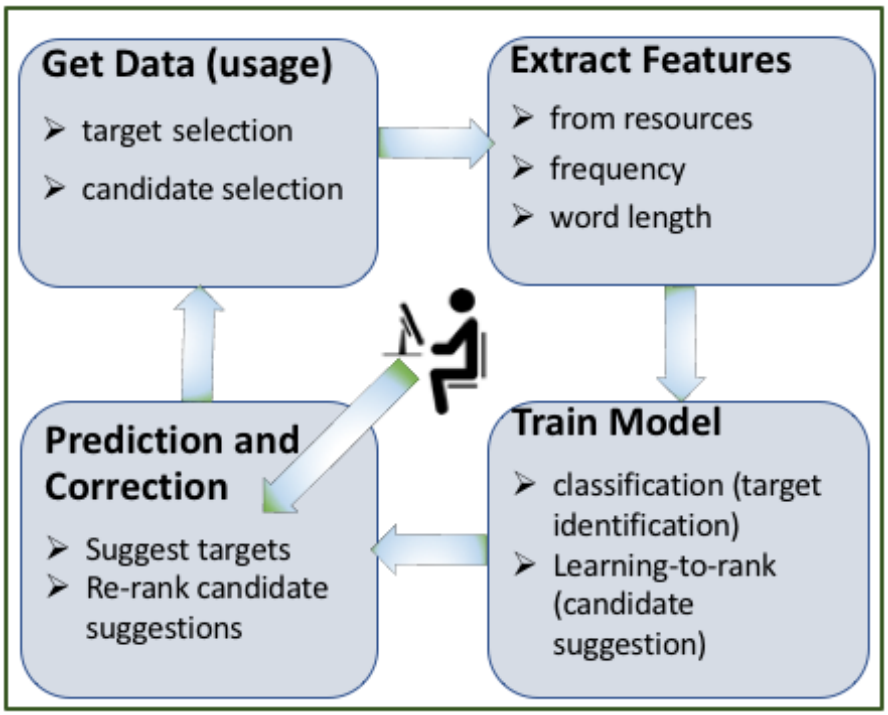


Figure 4.4.3: The loop for the generation of the adaptive models of **Par4Sem**.

The user interacts with the tool, probably accepting or rejecting the suggestions, which is considered as a training signal for the next iteration's model. Figure 4.4.4 shows the entirety of interactions, iterations, and adaptive processes of the Par4Sem system. In the first iteration, the ranking is provided using a baseline language model while for the subsequent iterations, the usage data from the previous batches (from batch 1 to batch t-1) is used to train a model that is used to rank the current batch (t).

4.4.1.3 Backend Technologies

The backend components are fully implemented using the Java programming language. The natural language pre-processing operations, such as sentence splitting, tokenization, lemmatization, and part of speech tagging are handled using the Apache OpenNLP[6] library.

[6] https://opennlp.apache.org/

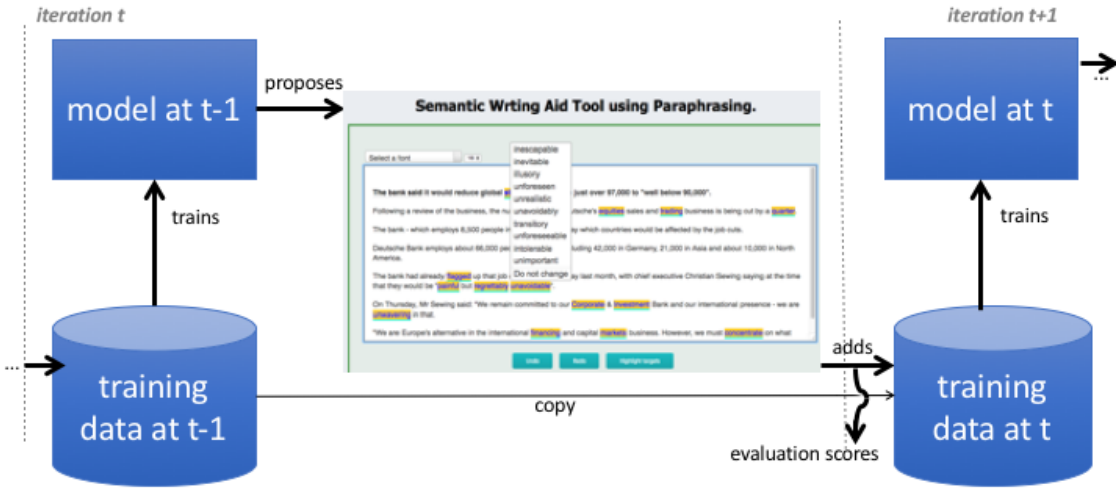


Figure 4.4.4: The iterative and adaptive interaction of **Par4Sem**.

For the target unit identification system, we have used *Datumbox*[7], a powerful open-source machine learning framework written in Java. Specifically, we have used the *Adaboost* classification algorithm in combination with the Multinomial Naive Bayes training parameters.

For the ranking model, RankLib[8], which is the popular library for the learning to rank algorithms from the Lemur project is integrated. All the data related to Par4Sem interactions (usage data, task completion timestamps, and user details) are stored in a MySQL database.

4.4.2 Frontend Components

The frontend component of Par4Sem is designed to facilitate document composing with a semantic paraphrasing capability. It is a web-based application, accessible from a browser either from a local installation or over the internet.

4.4.2.1 User Interface Components for Paraphrasing

The frontend component of Par4Sem comprises different modules. The most important user interface (UI) component is the text editing interface (Figure 4.4.5) that allows text production (writing from scratch or copy and paste from any external sources in different formats, such as

[7] http://www.datumbox.com/
[8] https://sourceforge.net/p/lemur/wiki/RankLib/

text, HTML, and word processing documents with their formatting), highlighting target units with different colors, and displaying candidate suggestions using drop-down UI component. (1) is the main area to compose (or paste) texts.

The operational buttons ((2)) are used to perform actions such as undo and redo (for writing, target unit highlighting, and paraphrase ranking), automatically highlighting target units, and clearing the text area. Target units are underlined in cyan color and highlighted as a yellow background in the form of hyperlink ((3)), which enables users to click, display, and select candidate suggestions for a replacement ((4)).

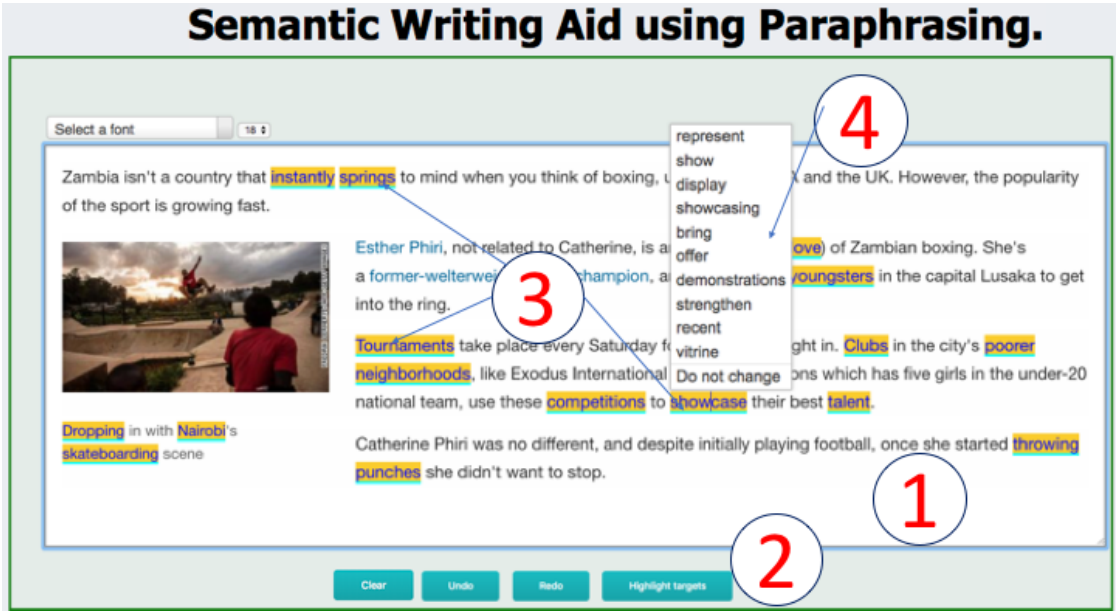


Figure 4.4.5: The **Par4Sem** text editing component that is used to compose texts, highlight target units, and display candidate suggestions for the target units.

4.4.2.2 Frontend Technologies

The frontend components are implemented using HTML, CSS, and JavaScript technologies. For the text highlighting and candidate suggestion replacement, the jQuery Spellchecker[9] module is slightly modified to incorporate the semantic highlighting (underline in cyan and a yel-

[9] http://jquery-spellchecker.badsyntax.co/

low background). The accompanied documentation and datasets of Par4Sem are hosted at the projects GitHub page[10].

4.4.3 RESTful API Component

Semantic technologies, those like Par4Sem incorporates highly dynamic parameters. One dimension is that the paraphrase resources can be different based on the requirements of the NLP application. Another aspect is that the application can be in different natural languages. If the backend and the frontend technologies are highly coupled, it will be difficult to reuse the application for different languages, resources, and applications. To overcome this problem, we have developed Par4Sem using a RESTful API (aka. microservices) as a middleware between the backend and the frontend components.

The API component consumes requests (getting target units and candidate suggestions) or resources (saving usage data such as selection of new target units, user's preference or selection for the candidate ranking, user and device information) from the frontend and transfers them to the backend. The backend component resolves the requests or resources and handles them accordingly. Spring Boot[11] is used to build the API services.

4.4.3.1 Installation and Deployment

As Par4Sem consists of different technologies, machine learning setups, resources, and configurations, we opted to provide Docker-based installation and deployment options. While it is possible to install the tool on one's server fully, we also offer API access for the whole backend service. The API service allows users to quickly and easily install the frontend component and rely on the API service calls for the rest of the communications.

[10] https://uhh-lt.github.io/par4sem/
[11] https://projects.spring.io/spring-boot/

Kill two birds with one stone!

English quote

5

Adaptive Visualization Tools to Collect Training Datasets

5.1 Visualization and Annotation

In Chapter 3 and Chapter 4, we have presented how datasets are collected using dedicated annotation tools or how datasets are obtained from usage data using an embedded model for adaptive NLP applications. Let us recall from the previous chapters that annotation tools are developed primarily to collect training datasets, where the datasets are used to build machine learning models. In the case of adaptive NLP applications, they relied on **usage data** to build the adaptive machine learning model.

However, **visualization tools** are primarily developed to present information using different information visualization components such as graphs, charts, and tables to make information exploration easier and faster.

Usually, for NLP-supported data visualization applications, the visualization tools include a machine learning model, for example, to automatically detect entities, relationships, and events. Hence, NLP-supported visualization applications depend on different NLP components, such as named entity recognition, topic and entity clustering, sentiment analysis, and event and relation detection. Some of the NLP-supported visualization tools we have developed and investigated their impact include a visualization tool for data journalists (**New/s/leak**), a visualization tool for network-based news items exploration (**Network of the day**), and a visualization tool that supports anonymization of legal court decisions (**AnonML**).

5.2 Network of the Day

Network of the day (NoD)[1] is a visualization tool, which helps users to discover essential topics released from different news providers daily more quickly. It presents networks of entities using a graphical visualization mainly focusing on automatically revealing relations between two or more entities and grouping or clustering entities based on their relationship.

5.2.1 Architecture of NoD

The architecture of NoD is shown in Figure 5.2.1. NoD relies for the source data on a crawler, which downloads the news feeds that are published every day. News feeds are crawled every half an hour, and the network graphs are built once in a day so that it is possible to discover insightful highlights of the day. Once news feeds are crawled, we run multiple NLP preprocessing steps such as splitting documents into appropriate sentences, extracting named entities (for example **Persons** and **Organizations**), and build relationship graphs based on entity co-occurrences.

5.2.2 User Interfaces of NoD

As it can be seen in Figure 5.2.2, NoD presents many interactive UI components. (1) shows the interactive and time-dependent network view of a given cluster, which is organized by a topic. Nodes represent named entities such as persons and organizations, and the edges between the nodes show the relations between the entities.

[1] http://www.tagesnetzwerk.de/

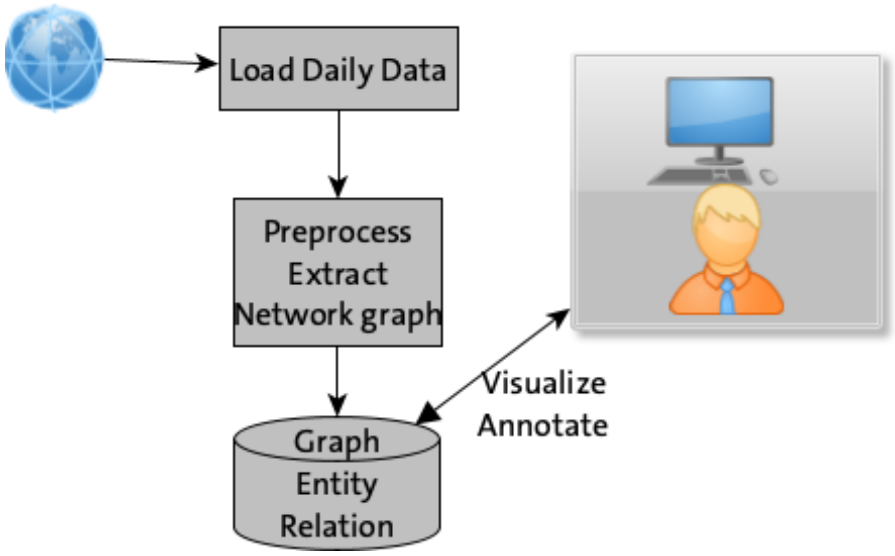


Figure 5.2.1: Architecture of Network of the day (NoD).

The frequency chart (2) shows 1) entities that are popular on the respective days and 2) trends of the entity and its relation with other entities over a period of time. What is more relevant in the scope of this work is the cooperative social tagging interface (3), which shows the source sentences for the entities with a link to the original online article and a text field to add a label (6) of entity relations. The tool also supports searching for entities (4) and viewing a specific network on another date with a date picker component (5)

5.2.3 Relation Annotation for Network Graphs

The social tagging component (Figure 5.2.2) allows adding the relation types between two entities. Users can read the source document and can make the relationship between two entities explicit. Moreover, once the relation type is annotated for a given day, the tool automatically recommends the same annotation for similar entities discovered in the dataset.

The relation annotation also allows to find out if the two entities maintain the same relationship over time or not, which is displayed in the relationship frequencies chart (see 2 in Figure 5.2.2). In the future, the relation annotation collected can be used to train a relation tagging machine learning model, similar to the IBM's statistical information and relation extraction

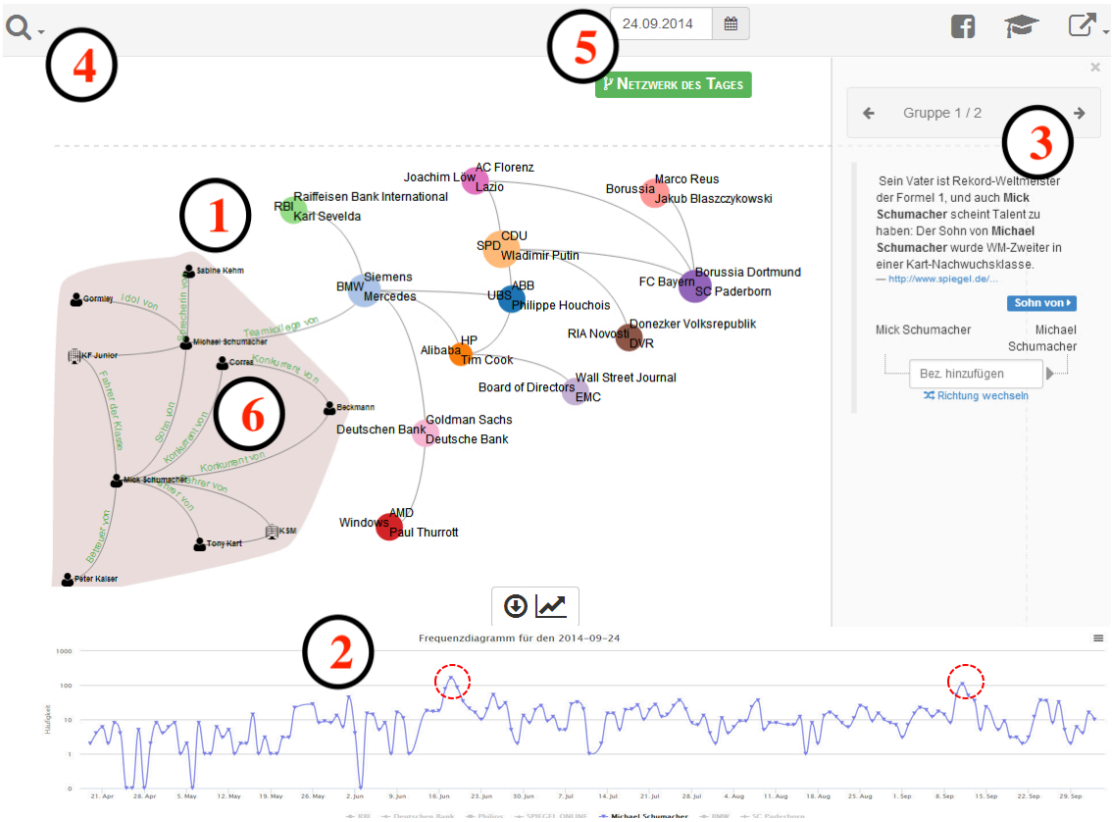


Figure 5.2.2: UI component of (NoD).

(SIRE)[2] system.

5.3 New/s/leak

New/s/leak[3] (Wiedemann et al., 2018a,b; **Yimam** et al., 2016c), the *network of searchable leaks*, is a journalistic software to investigate and visualize large textual datasets. Investigation of unstructured document collections is a difficult task: The sheer amount of content can be overwhelming, for instance, the WikiLeaks PlusD[4] dataset contains around 250 thousand diplomatic cables. Typically, these collections mostly consist of unstructured text with additional

[2] https://researcher.watson.ibm.com/researcher/view_group.php?id=2223
[3] http://newsleak.io
[4] https://wikileaks.org/plusd/about

metadata such as date, location or sender and receiver of messages. The most substantial part of these documents is irrelevant for journalistic investigations, concealing the crucial storylines. For instance, war crime stories in WikiLeaks were hidden and scattered among hundreds of thousands of routine conversations between officials. Therefore, if journalists do not know in advance what to look for in the document collections, they can only vaguely target all people and organizations (named entities) of public interest.

Although New/s/leak primarily is used to help investigative journalists to uncover hidden facts or stories, it also incorporates several NLP components that support the process. Instead of having a static NLP model, New/s/leak supports annotation (labeling new entities and correcting wrong entities) and tagging (adding a tag to the document and tagging keywords) functionalities to make the tool more interactive and dynamic during the investigative process. In the following subsections, we elaborate the main NLP components and the different types of annotation capabilities the tool supports.

5.3.1 System Architecture of New/s/leak

Figure 5.3.1 shows the overall architecture of **New/s/leak**. To allow users to analyze a wide range of document types, New/s/leak includes an internal document processing component, which extracts text and metadata from various document types, into a unified representation. On this unified text representation, a number of NLP pre-processing tasks are performed using the UIMA pipeline (Ferrucci and Lally, 2004), e.g., automatic identification of the document language, segmentation of documents into the appropriate paragraph, sentence and token units, and extraction of named entities, keywords, and metadata.

The backend employed ElasticSearch[5], a powerful document indexing and retrieval system, to store, retrieve, and perform aggregation operations. All the end-user requirements are implemented as RESTful web services based on the Scala Play[6] framework. State-of-the-art information visualization technologies are used, which are composed of the AngularJS browser app to present information to the journalists. Visualizations, among other things, are realized with the D3 library Bostock et al. (2011), include network graphs, bar charts, and text highlighting tools. To enable seamless deployment of the tool by journalists with limited technical knowledge, all

[5] https://www.elastic.co/
[6] https://www.playframework.com/

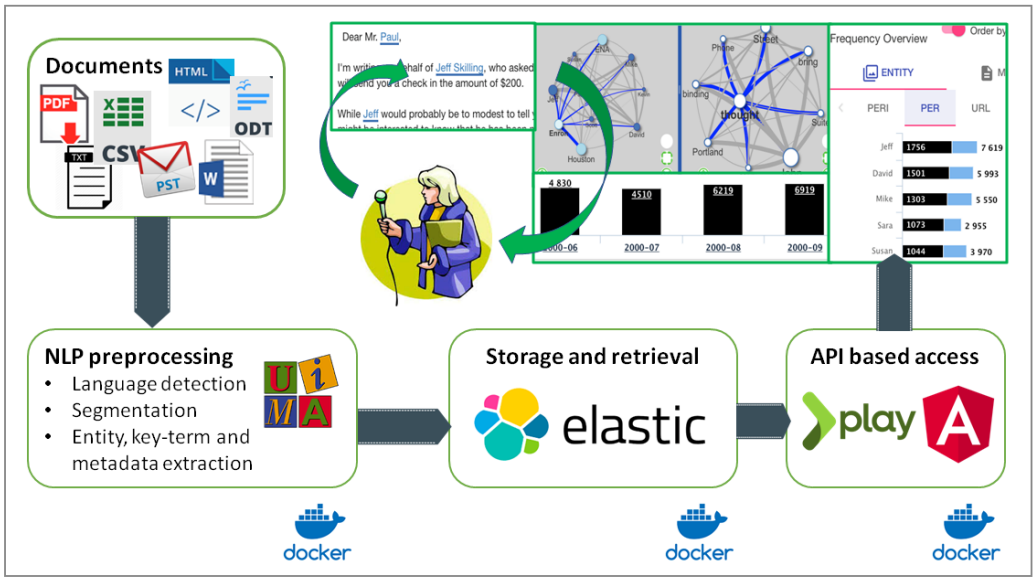


Figure 5.3.1: Architecture of *New/s/leak*

the required components of the architecture are integrated into a Docker[7] setup. Via docker-compose, a software to orchestrate Docker containers for complex architectures, end-users can download and run locally a preconfigured version of New/s/leak with one single command. Being able to process data locally and even without any connection to the internet is a vital prerequisite for journalists, especially if they work with sensitive leaks data. All necessary source code and installation instructions can be found on the GitHub project page[8].

5.3.2 Entity- and Keyword-Centric Visualization

Access to unstructured text collections via named entities is essential for journalistic investigations. To support this, two types of graph visualization are included, as it is shown in Figure 5.3.2. The first graph, called **entity network**, displays entities in a current document selection as nodes and their joint occurrence as edges between nodes. Different node colors represent dif-

[7] https://www.docker.com
[8] https://uhh-lt.github.io/newsleak/

ferent entity types such as a person, organization, or location names. Furthermore, mentions of entities that are annotated based on dictionary lists or annotated by a given regular expression are included in the entity network graph. The second graph, called the **keyword network**, is build based on the set of keywords representing the current document selection. The keyword network also includes tags that can be attached to documents by journalists while investigating the collection.

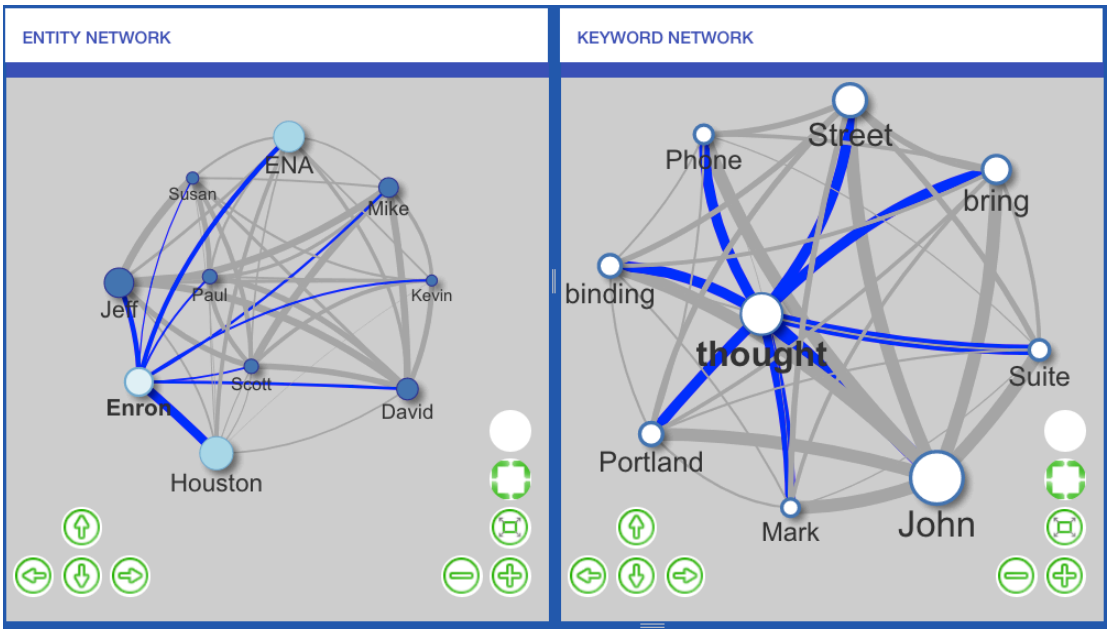


Figure 5.3.2: The entity and keyword graphs of new/s/ leak are based on the Enron email dataset (Keila and Skillicorn, 2005). Networks are visualized based on the current document selection, which can be filtered by full-text search, entities, or metadata. Visualization parameters such as the number of nodes per entity type or minimum edge strength can be set directly in the UI by the user. Edge colors highlight connections for currently selected nodes. Hovering over nodes and edges in one graph highlights connections between the respective graph to show which entities and keywords frequently co-occur with each other in documents.

The New/s/leak UI components are designed mainly to help journalists to discover interesting stories more quickly. Figure 5.3.3 presents different UI components that facilitate close reading[9]. The numbers in Figure 5.3.3 depict the different types of visual components sup-

[9]**Close reading** is a way of reading texts critically, where the reader pays special attention to understand the

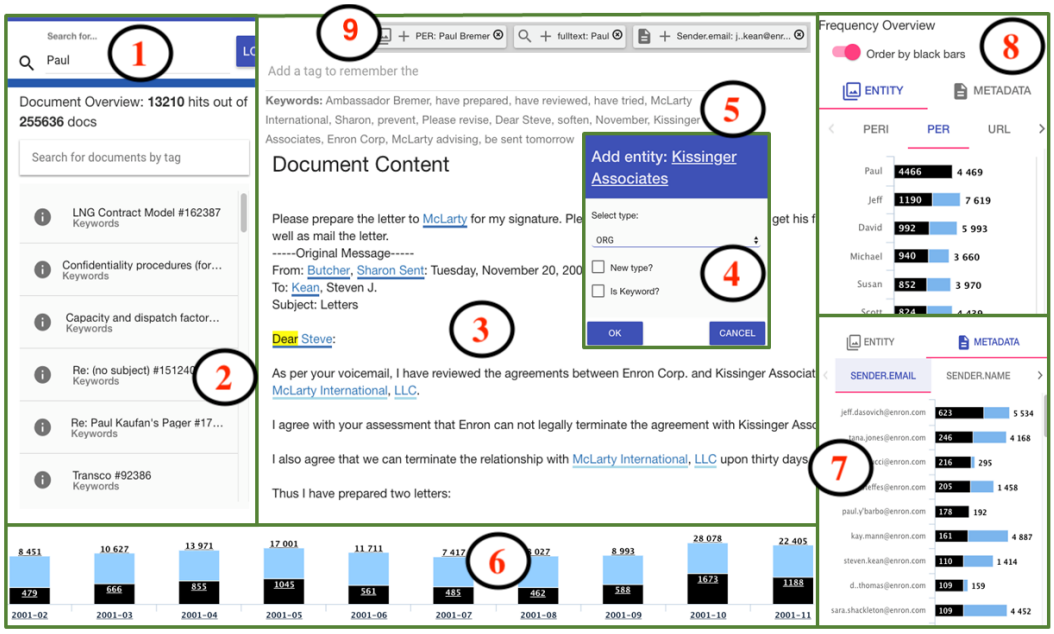


Figure 5.3.3: Different UI components of the *New/s/leak* system. The components can be activated or hidden based on the user's need and interactions.

ported in the tool.

(1) shows the full-text searching component, (2) presents the list of documents that are retrieved based on the search filter or other criteria for close reading. In (3), the user can read the actual document content, but also annotate new entities ((4)), which were not detected by the NER, merge different textual references to the same entities, and remove/blacklist wrongly recognized entities.

The possibility to tag or annotate individual documents can be used to organize the collection with respect to any user-defined category system. (5) shows list of keywords summarizing the document. (6), (7), and (8) display temporal evolution, metadata, and entity charts to help the reader to further filter documents. Lastly, (9) shows a history of search filters applied to the document collection so far.

details of a given topic.

5.3.3 Annotation Processes in New/s/leak

New/s/leak incorporates several annotation and correction strategies. If the named entity recognition (NER) component produces wrong predictions, it is possible to either 1) delete or blacklist the annotation if the annotation is entirely wrong (for example the annotation is a non-named entity object) or 2) correct the annotation by changing the entity type. The correction of entities could help to improve the performance of the embedded NER model.

New/s/leak also supports collaborative annotation of documents. The annotations or labels help to group similar documents in the same category. The journalists will share the annotated tags to filter interesting documents for close reading.

5.4 AnonML

AnonML is web-based software implemented to anonymize legal documents, primarily court decisions. Many countries do not allow publishing documents of court decisions without hiding legal entities such as name, location, or company of the offender. There is always a conflict between the demand to have the right of information (**open data**) and the right of individuals to ensure the confidentiality of private data (Vico and Calegari, 2015). In this context, publication of court decisions has been heavily hindered where there is a law in the country, such as Germany, that prevent publishing the name and related confidential information of the offenders publicly.

One way to publishing court decision is **anonymization**, which means that the name of the offender is replaced or abbreviated with arbitrary representations. We have developed AnonML, which is especially dealing with the process of anonymizing and publishing court decisions. Document anonymization is the process of replacing sensitive data with an arbitrary representation to ensure the confidentiality of sensitive data.

5.4.1 Design of AnonML

AnonML comprises different components. At the backend, there are document management, machine learning and rule-based annotation mechanisms, and a database component to store different system configurations (see Figure 5.4.1). The frontend component is responsible for

displaying and allowing an interactive process to anonymize legal documents. The primary objective of AnonML is to anonymize (replace sensitive entities with non-recognizable representation) entities or mentions related to the defendant such as the name, organization, contact (email and telephone), or address.

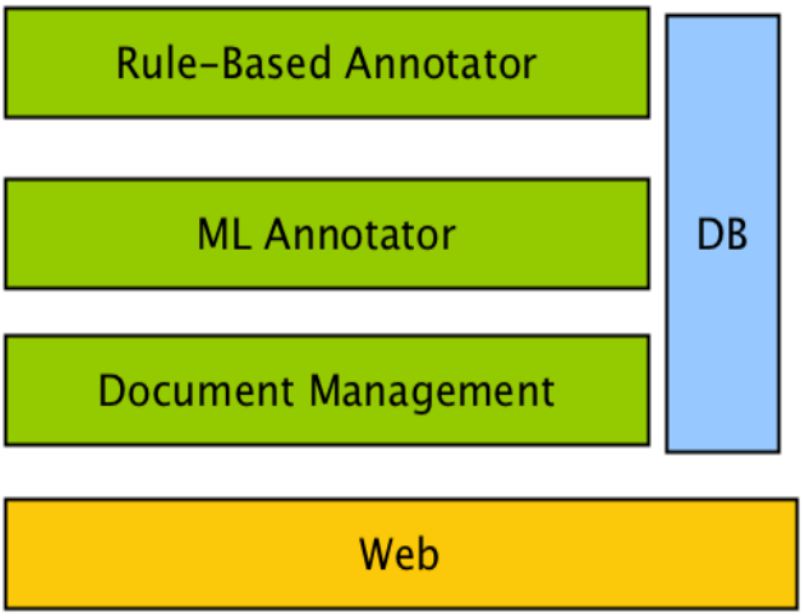


Figure 5.4.1: Architecture of AnonML.

5.4.2 Annotation and Adaption Processes

As we can see in Figure 5.4.2, entities detected by the system can be displayed in different color. The user, here the legal expert responsible for the anonymization of the document, can perform the following tasks.

1. **Accept system suggestion**: The user can press the **accept** button (with shortcut **'a'**) and the system will automatically replace the entity with a suggested code.

2. **Reject suggestions**: If the proposed entities are wrong, the user can discard the suggestion by pressing the **reject** button (with shortcut **'d'**).

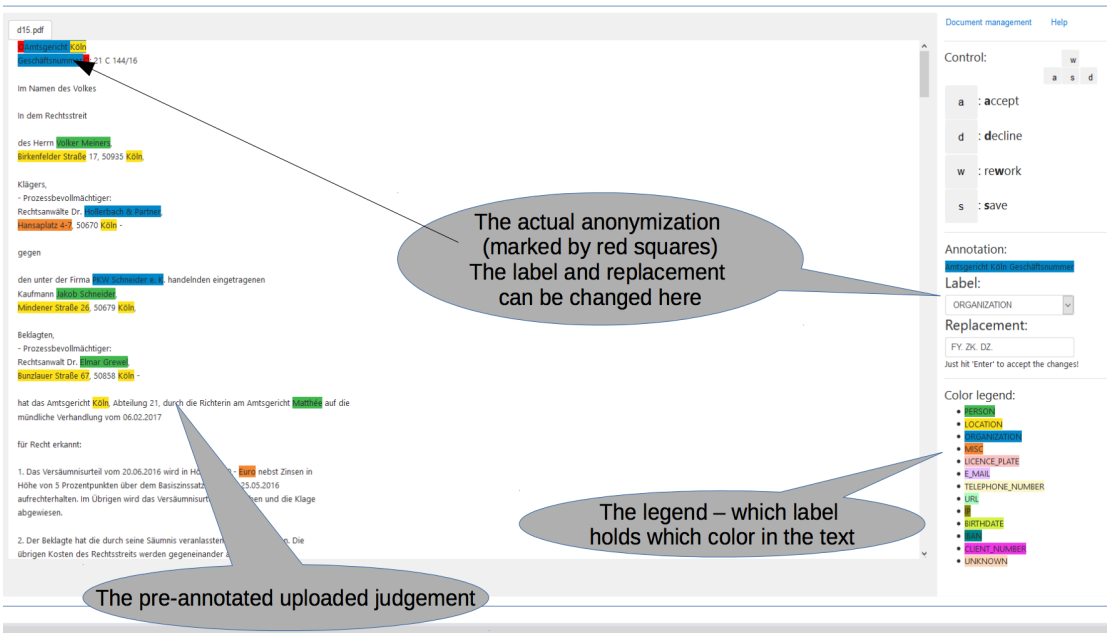


Figure 5.4.2: The user interface for the different anonymization steps in AnonML.

3. **Rework the suggestion**: It is also possible to rework or change the entity type if the suggestion is in a wrong category.

4. **Annotate own entity**: If the machine learning or the rule-based annotator fails to capture some entities, the user can select the entity and assign an appropriate category.

The goal of AnonML is hence to build an automatic system that can help in anonymizing court decisions with minimal human intervention. Moreover, AnonML also supports importing PDF documents, which are usually scanned court decisions, and produce the anonymized version into another PDF document.

AnonML can be easily deployed with a docker system to avoid the need for IT experts for the installation and configuration works. Furthermore, since AnonML is a web-based tool, it can be easily used by users from their browser. To further restrict access only to trusted users, it can be either installed in a private local area network or can be protected with additional password-based authentication.

5.4.3 Automation Components of AnonML

We have considered the anonymization problem as a named entity recognition problem. Every named entity found in the document is deemed to be sensitive to be anonymized. In this regard, we have integrated GermaNER (Benikova et al., 2015), a publicly available named entity recognizer (NER) for German that we have developed during the thesis work. Besides the NER model, we have also developed rules and patterns that can automatically capture legal entities such as car plate numbers, area codes, phone numbers, and email addresses.

The NER model and the rules are used to generate initial suggestions for the anonymization. Based on these suggestions, the legal expert can continue with the anonymization process, i.e., accepting or rejecting the suggestions.

5.4.4 Automatic Anonymization and Adaption

Figure 5.4.2 shows the user interface of AnonML. The tool provides different options for the anonymization of legal documents. For entities that are automatically identified by the NER model or set of rules, the expert can either accept or decline the anonymization. If there are entities that are not detected by the model, it is also possible to manually mark the entities for anonymization. To facilitate the rapid annotation process, AnonML also supports **repeat anonymization**, which looks for the same entities in the document and anonymizes them with the same code.

Once the process of anonymization is completed, it is possible to export the anonymized document into a **PDF** or **HTML** file format that can be published publicly. But most importantly, the user feedback is directly fed back to the NER system to re-train a better anonymization model. The generation of a new training model helps to improve the anonymization system, which also dramatically reduces the manual anonymization effort.

It is quite evident that this adaptive nature of anonymization is an essential requirement instead of integrating a static NER model. As we can see, the NER model is built with a general purpose NER system (GermaNER). However, the model is getting more personalized and adaptive based on user feedback. AnonML also has an option to train the new model exclusively from the new training examples collected during anonymization process, in this example, completely avoiding the GermaNER for a further anonymization task.

We have also learned that the anonymization tool can be used for a different domain than the legal field, such as for medical documents. Most of the medical records manifest sensitive data, which require the same anonymization process as the legal documents. Even though we can use the same NER model initially for different applications, the adaptive system helps to generate a specialized model based on the essence of the application and the user feedback collected during the anonymization process.

5.5 Conclusion

In this chapter, we have briefly illustrated how to collect training examples using NLP-supported visualization tools. NLP-supported visualization tools differ from conventional NLP applications as the focus of the former is mainly on presenting information in an easy to understand visualization interface. Visualization tools are also different from annotation tools, primarily as they are not purposely developed to collect training datasets.

The NLP-supported visualization tools require an initial dataset to train a model (that might not be good enough in performance) and present the expected visualization output. The advantage of collecting training examples while utilizing the visualization tool is two-fold: 1) we can collect training examples while using the tools, and 2) it is possible to re-train the existing NLP model of the visualization tools, which further makes the tool better over time based on the feedback. Furthermore, the data collected can even be used to train models for different NLP applications.

Part III

Resources for Semantic-aware NLP Applications

Semantic-aware NLP applications require different resources, which should be collected using different annotation tools. In this part of the thesis, we first discuss semantic-aware NLP resources collected using annotation tools along with corresponding machine learning models and experimental results. We then discuss resources that are obtained during the development of an adaptive NLP application.

In Chapter 6, we discuss how the paraphrase resources are collected, compiled, and evaluated for a semantic writing aid tool that incorporates a paraphrasing component. In Chapter 7, we explain the different approaches we have followed to collect complex phrase identification datasets, which often are a prerequisite task for text simplification.

Original: He has tons of stuff to throw away.
Paraphrase: He needs to get rid of a lot of junk.

Examples of Paraphrasing[a]

[a] http://bit.do/paraphrase-example

6 Building Paraphrase Resources

In this chapter, we present our findings on the collection of paraphrase resources and the impact of context for the lexical paraphrase ranking task comparing and quantifying results for multi-word expressions and single words. We focus on the systematic integration of existing paraphrase resources to produce candidate paraphrases and later ask human annotators to judge paraphrasability/substitutability in context.

6.1 Context-Sensitive Paraphrase Collection

Here, the primary objective is to collect paraphrase resources by presenting some context, for example, previous and next sentences, and examine the influence of contexts particularly for paraphrasing of multi-word expressions (MWEs) (**Yimam** et al., 2016b). Paraphrasing is an alternative way of writing texts while conveying the same information (Burrows et al., 2013; Zhao et al., 2007). Several semantic-aware NLP applications, such as text shortening (Burrows

et al., 2013), text simplification, machine translation (Kauchak and Barzilay, 2006), or textual entailment (Androutsopoulos and Malakasiotis, 2010) require an automatic text paraphrasing component.

Over the last decade, a large number of static paraphrase resources have been developed. Static paraphrase resources do not include contextual information. The paraphrase database (PPDB) (Pavlick et al., 2015) is the most prominent and the largest paraphrase resource available to date. However, PPDB provides only paraphrases without context. The lack of context in PPDB impedes the usage of such a resource in context-aware NLP applications.

In this respect, we tackle the problem of automatically ranking candidate paraphrases based on context (contextualization of paraphrases) using abundantly available paraphrase resources. Furthermore, we focus on multi-word paraphrases, since single-word replacements are covered well in lexical substitution datasets, such as Biemann (2012); McCarthy and Navigli (2007). While most of the existing datasets contain multi-word substitution candidates, the substitution targets are always single words. Multi-word expressions are prevalent in a text, constituting roughly as many entries as single words in a speaker's lexicon (Sag et al., 2002), and are essential for many NLP applications. For example, the work by Finlayson and Kulkarni (2011) shows that the detection of multi-word expressions improves the F-score of a word sense disambiguation task by 5 percent. We experiment with both MWE and single words and investigate the difficulty of the paraphrasing task for single words vs. MWEs, using contextual features.

For the development of the context-aware paraphrase resources, we follow these steps: 1) systematic combination of existing paraphrase resources to produce candidate paraphrases for single- and multi-word expressions, 2) collection of a dataset for a paraphrase ranking/selection annotation task using crowdsourcing, and 3) evaluating and investigating different machine learning approaches for an automatic paraphrase ranking.

Below, we define technical terms relevant to paraphrase resources.

Target: A **target** is a word or MWE in a text (sentence or paragraph) that is going to be paraphrased (see Chapter 4, Figure 4.2.1).

Candidate paraphrase: Candidate paraphrase is a possible substitute for a target word or MWE in a text.

6.2 Related Work

6.2.1 Paraphrase Resources and Machine Learning Approaches

Paraphrasing consists of mainly two tasks, paraphrase **generation** and paraphrase **identification**. Paraphrase generation is the task of obtaining candidate paraphrases for a given target. Paraphrase identification estimates whether a given candidate paraphrase can replace a target without changing the meaning in context.

PPDB (Pavlick et al., 2015) is one of the largest collections of paraphrase resources collected from bilingual parallel corpora. PPDB2 has recently been released with revised ranking scores. It is based on human judgments for 26,455 paraphrase pairs sampled from PPDB1. Ridge regression is applied to rank paraphrases, using the features from PPDB1 and include word embeddings.

The work of Kozareva and Montoyo (2006) uses a dataset of paraphrases that were generated using monolingual machine translation. In the dataset, sentence pairs are annotated as being a paraphrase or not. For the binary classification, they use three machine learning algorithms support vector machine (SVM), k-nearest neighbors (kNN), and, maximum entropy (MaxEnt). As machine learning features, they use word overlap features, n-grams ratios between targets and candidates, skip-grams longest common subsequences, POS tags, and proper names.

Connor and Roth (2007) develop a global classifier that takes a word v and its context, along with a candidate word u, and determines whether u can replace v in the given context while maintaining the original meaning. Their work focuses on verb paraphrasing. Notions of context include: to be either subject or object of the verb, named entities that appear as subject or object, all dependency links connected to the target, all noun phrases in sentences containing the target, or all of the above.

The work of Brockett and Dolan (2005) uses annotated datasets and Support Vector Machines (SVMs) to induce larger monolingual paraphrase corpora from a comparable corpus of news clusters found on the World Wide Web. Features include morphological variants, WordNet synonyms and hypernyms, log-likelihood-based based word pairings dynamically obtained from baseline sentence alignments, and string features such as word-based edit distance

Using a dependency-based context-sensitive vector-space approach, Thater et al. (2009) com-

pute vector-space representations of predicate meaning in context for the task of paraphrase ranking. An evaluation on a subset of the SemEval 2007 lexical substitution task produces a better result than previous state-of-the-art systems.

Zhao et al. (2007) address the problem of context-specific lexical paraphrasing using different approaches. First, similar sentences are extracted from the web and candidates are generated based on syntactic similarities. Candidate paraphrases are further filtered using POS tagging. Second, candidate paraphrases are validated using different similarity measures such as co-occurrence similarity and syntactic similarity.

Our approach is similar to previous works on all-words lexical substitution (Hintz and Biemann, 2016; Kremer et al., 2014; Szarvas et al., 2013) in the sense that we construct delexicalized classifiers for ranking paraphrases: targets, candidate paraphrases, and context are represented without lexical information, which allows us to learn a single classifier/ranker for all potential paraphrasing candidates. However, the existing approaches are limited to single-word targets (Szarvas et al., 2013) respectively single-word substitutions (Kremer et al., 2014) only. In our approach, we have extended these notions to MWE targets and candidates. We also investigate the differences with the single-word approaches and report both on classification and ranking experiments.

6.2.2 Multi-word Expression Resources

Various approaches exist for the extraction of MWEs: Tsvetkov and Wintner (2010) present an approach to extract MWEs from parallel corpora. They align the parallel corpus and focus on misalignment, which typically indicates expressions in the source language that are translated to the target in a non-compositional way. Frantzi et al. (2000) present a method to extract multi-word terms from English corpora, which combines linguistic and statistical information. The Multi-word Expression Toolkit (*MWEtoolkit*) extracts MWE candidates based on flat n-grams or specific morphosyntactic patterns (of surface forms, lemmas, POS tags) (Ramisch et al., 2010) and apply different filters ranging from simple count thresholds to a more complicated case such as association measures (AMs). The tool further supports indexing and searching of MWEs, validation, and annotation.

Schneider et al. (2014) developed a sequence-tagging-based supervised approach to MWE identification. The model was built with a lot of features such as word unigrams and bigrams, character prefixes and suffixes, POS tags, and different MWE lexicons constructed from from various sources. The work by Vincze et al. (2011) constructs a multi-word expression corpus annotated with different types of MWEs such as a compound, idiom, verb-particle constructions, light verb constructions, and others. In our approach, we have used a combination of multiple MWE resources from different sources for both MWE target detection and candidate generation (see Subsection 6.3.2). There are no previous works related to particularly paraphrasing multi-word expressions.

6.3 Methods to Collect Datasets

The paraphrase resources are collected using the annotation tool we discuss in Section 4.2.

6.3.1 Impact of Context on Paraphrasing

To demonstrate that contexts have influences on paraphrasing, we conduct different experiments using the **PPDB2** paraphrase database. **PPDB2** is released with better paraphrase ranking than PPDB1 (Pavlick et al., 2015) but does not incorporate context information.

For this task, a total of 171 sentences are selected from the British Academic Written English (BAWE) corpus[1] (Alsop and Nesi, 2009). For each HIT (3–8 sentences), 5 targets are selected. MWEs are considered as targets when they have at least 5 candidate paraphrases from the PPDB2 resource. The workers can select up to three paraphrases and have to supply their own paraphrase if none of the candidates fits the context.

We conduct this annotation task in two setups, the first experiment by showing the original context (3–8 sentences) of the targets and the second experiment only by showing the targets and the candidates without contexts. For both setups, a task is assigned to 5 MTurk workers. We incorporate control questions with invalid candidate paraphrases to reject unreliable workers. **Control questions** in MTurk are one way of screening out workers, where questions include purposely wrong answers (in our case, adding candidates, which are unrelated to the target word

[1] https://www2.warwick.ac.uk/fac/soc/al/research/collections/bawe/

	All (ρ)	MWE (ρ)	Single (ρ)
No context	0.35	0.25	0.36
Context	0.31	0.23	0.32

Table 6.3.1: Spearman correlation of human annotation with PPDB2 default rankings. The column *MWE* shows the result of only MWEs and the column *Single* shows the result of only single words.

or phrase by any means) to see if workers are doing the task with proper attention or they really understand the task at all. These HITs are excluded for the experiments.

In addition to the control questions, JavaScript functions are embedded to ensure that workers either select one candidate paraphrase or provide their own candidate paraphrase.

The rank of the candidate paraphrase is computed by summing the number of workers those agreed on the candidate, for scores between 0 (no worker chooses the candidate paraphrase) and 5 (all the workers agree on the candidate paraphrase). Table 6.3.1 shows the Spearman correlation results between human judgment and PPDB2 default rankings of the target and the candidate paraphrase. We can see that both single and MWE targets are context-dependent, as correlations are consistently lower when taking context into account. Further, we note that correlations are positive, but low, implying that the PPDB2 ranking should not be used as-is for paraphrasing.

6.3.2 Paraphrase Dataset Collection using Crowdsourcing

In this subsection, we present the processes carried out to collect datasets for the paraphrase ranking task. **Paraphrase ranking** is an essential sub-task for a number of semantic aware NLP applications, whose objective is to rank candidate paraphrase substitutes based on the requirements of the application.

The collection of the resource includes the selection of documents, the identification of targets for paraphrasing, and the generation of candidate paraphrases from existing resources. We use 2.8k essay sentences from the ANC[2] and BAWE corpora for the annotation task.

Target detection: To investigate the impact of contexts for paraphrasing, the first step is to determine possible targets for paraphrasing (see Figure 4.2.1 in Chapter 4). In fact, every word

[2] http://www.anc.org/

or MWE in a sentence can be considered as a target for paraphrasing. When prototyping the annotation setup, we found that five targets are a reasonable amount to be completed in a single **Human Intelligence Task** (HIT - a single and self-contained unit of a task in MTurk to be completed and submitted by a worker to receive a reward in return)[3].

We select targets that possess at least five candidates in our combined paraphrase resources. The paraphrase resources (S) for candidate generation are composed from collections of PPDB (Pavlick et al., 2015), WordNet, and JoBimText distributional thesaurus (DT – only for single words).

For MWE targets, we combine different MWE resources. First, a total of 79,349 MWE are collected from **WordNet**, **STREUSLE** (Schneider and Smith, 2015; Schneider et al., 2014)[4], **Wiki50** (Vincze et al., 2011), and **the MWE project** (Baldwin and Villavicencio, 2002; McCarthy et al., 2003)[5]. Second, we consider MWEs from these resources to be a target when it is possible to generate candidate paraphrases from our paraphrase resources (S). A better and adaptive approach of detecting targets will be discussed in Chapter 9, in the context of developing an adaptive semantic writing aid tool.

Candidate generation: Candidate paraphrases for a target (both single and MWE) are generated as follows. For each target, we retrieve candidates from the resources (S). When more than five candidate paraphrases are obtained: 1) for single words, we select the top candidates that exhibit differences in meanings of context using the automatic sense induction API (Ruppert et al., 2015), 2) for MWEs we choose candidates that are obtained from multiple resources in S, i.e, the more resources the candidate appears in, the higher the chance it is regarded as a possible candidate.

We present five candidates for the workers to choose the suitable candidates in context. We also allow workers to provide their own alternative candidates when none of the provided candidates fits the current context. We discuss the details of the annotations collected in Section 6.6.2. An adaptive approach for candidate re-ranking for the semantic writing aid application will be discussed in Chapter 9.

[3] https://www.mturk.com/mturk/help?helpPage=overview
[4] http://www.cs.cmu.edu/~ark/LexSem/
[5] http://mwe.stanford.edu

	kNN			LambdaMART		
Features	P	R	F	P@1	NDCG@5	MAP
All	69.27	90.41	78.41	90.53	89.03	**91.35**
F0+1+2+5	76.14	84.40	80.04	89.38	89.24	91.31
F1+2	75.28	85.05	79.85	88.13	88.98	90.88
F1+3	75.28	85.05	79.85	88.13	88.98	90.88
F1+5	74.42	86.69	**80.07**	88.11	88.76	90.82
F0+1+2+7	74.89	85.65	79.89	89.42	89.34	91.29
F3+7	70.28	79.82	74.61	82.31	84.08	86.34
F5+7	64.56	86.25	73.64	80.24	82.61	85.60
F0+3	68.87	81.39	74.43	87.04	86.37	88.78
F0+7	69.86	79.02	74.05	84.14	84.69	87.20
F6+7	65.20	79.49	71.34	80.03	84.98	85.54
F0+6	67.43	78.04	72.08	84.98	85.26	87.64
F0	72.49	79.84	75.18	84.12	84.51	87.15

(a) Performance on all datasets

	kNN			LambdaMART		
Features	P	R	F	P@1	NDCG@5	MAP
All	76.74	82.99	79.71	89.72	88.82	**91.58**
F0+1+2+5	75.36	84.54	79.67	90.38	89.10	91.41
F1+2	75.74	83.66	79.49	88.28	88.82	90.98
F1+3	75.74	83.66	79.49	88.28	88.82	90.98
F1+5	74.95	85.52	**79.87**	87.50	88.51	90.76
F0+1+2+7	69.59	88.63	77.95	90.00	89.31	91.49
F3+7	70.25	78.71	74.09	81.92	83.78	86.03
F5+7	64.05	85.20	72.90	79.96	82.24	85.09
F0+3	68.89	80.52	74.05	86.41	86.46	88.64
F0+7	69.93	78.38	73.77	84.14	84.77	87.11
F6+7	64.67	78.80	70.71	78.97	82.06	84.98
F0+6	66.98	77.28	71.44	85.21	85.04	87.55
F0	74.08	72.18	71.47	84.81	84.60	87.29

(b) Performance on single words datasets

	kNN			LambdaMART		
Features	P	R	F	P@1	NDCG@5	MAP
All	69.81	95.70	80.60	84.69	77.54	86.21
F0+1+2+5	73.66	91.25	**81.56**	81.76	76.40	85.43
F1+2	73.25	91.11	81.13	82.74	76.00	86.69
F1+3	73.25	91.11	81.13	82.74	76.00	86.69
F1+5	72.58	92.05	81.05	84.69	77.14	**87.14**
F0+1+2+7	72.85	91.14	80.89	83.71	75.95	84.97
F3+7	71.56	85.18	77.57	78.83	72.71	80.40
F5+7	68.03	89.72	77.18	72.31	67.27	80.66
F0+3	70.05	85.64	76.91	81.43	71.32	81.62
F0+7	70.28	84.56	76.56	71.34	67.76	77.35
F6+7	69.46	85.38	76.45	79.48	67.82	79.66
F0+6	71.49	82.35	76.39	80.78	69.16	82.37
F0	73.35	70.54	69.06	69.71	67.12	77.95

(c) Performance on MWEs datasets

Table 6.3.2: Binary classification vs. learning to rank results on baseline and 8 top-performing feature combinations.

6.4 Machine Learning Approaches for Paraphrasing

The paraphrase selection problem is tackled by means of two different approaches. In the first approach, we build a **binary classification** model, which exclusively decides if a given candidate is an accurate replacement/paraphrase or not. To train the model, we consider all candidates that are provided by the worker as positive examples while the remaining of the candidates are taken as negative training examples.

In the second setup, we use a **learning to rank** algorithm to re-rank candidate paraphrases. There are different machine learning methods for the learning to ranking approach, such as *pointwise*, *pairwise* and *listwise* rankings. In pointwise ranking, a model is trained to map candidate phrases to relevance scores, for example using a simple regression technique. A ranking is then performed by simply sorting predicted scores (Li et al., 2007). In pairwise approach, the problem is deemed as a binary classification task where pairs are individually compared with each other (Freund et al., 2003). Listwise ranking approaches learn a function by taking individual candidates as instances and optimizing a loss function defined on the predicted instances (Xia et al., 2008).

We experiment with different learning to rank algorithms [6] from the RankLib[7] Java package of the Lemur project[8]. Here, we present the results obtained using the LambdaMART learning to rank algorithm. LambdaMART (Burges, 2010) uses gradient boosting to directly optimize learning to rank specific cost functions such as Normalized Discounted Cumulative Gain (NDCG) and Mean Average Precision (MAP).

6.4.1 Machine Learning Features

We have modeled three types of features: a **resource-based** feature where feature values are taken from a lexical resource ($F0$), four features based on **global context** where we use word embeddings to characterize targets and candidates irrespectively of context ($F1, F2, F3, F4$) and four features based on **local context** that take the relation of target and candidate with the context into account ($F5, F6, F7, F8$).

[6] We tried all the 8 available algorithms and LambdaMART performs better than others.
[7] https://people.cs.umass.edu/~vdang/ranklib.html
[8] http://sourceforge.net/projects/lemur/

PPDB2 score: The PPDB2 score (F_0) of each candidate is used as a baseline feature. This score reflects a context-insensitive ranking as provided by the lexical resources.

Let us first describe features considering global context information:

Target and Candidate phrases: We do not use word identity as a feature, but use the word embedding instead for the sake of robustness. We use the word2vec python implementation of Gensim (Řehůřek and Sojka, 2010)[9] to generate embeddings from BNC[10], Wikipedia, BAWE and ANC. We train embeddings with 200 dimensions using skip-gram training and a window size of 5. We approximate MWE embeddings by averaging the embeddings of their parts. We use the word embeddings of the target (F_1) and the candidate (F_2) phrases.

Candidate-Target similarities: The dot product of the target and candidate embeddings (F_3), as described in Melamud et al. (2015).

Target-Sentence similarity: The dot product between a candidate and the sentence, i.e., the average embeddings of all words in the sentence (F_4).

The following features use local context information:

Target-Close context similarity: The dot product between the candidate and the left and right 3-gram (F_5) and 5-gram embedding (F_6) respectively.

Ngram features: A normalized frequency for a 2- to 5-gram context with the target and candidate phrases (F_7) based on Google Web 1T 5-Grams[11].

Language model score: A normalized language model score using a sentence as a context for the target and candidate phrases (F_8). An n-gram language model (Kneser and Ney, 1995) is built using the BNC and Wikipedia corpora.

6.5 Experimental Results

We discuss the different experimental results using the K-Nearest Neighbors (kNN)[12] from the scikit-learn[13] machine learning framework (for the binary classification setup) and the LambdaMART learning to rank algorithm from RankLib (for the learning to rank setup).

[9] https://radimrehurek.com/gensim/models/word2vec.html
[10] http://www.natcorp.ox.ac.uk/
[11] https://catalog.ldc.upenn.edu/LDC2009T25
[12] Parameters: Number of neighbors (n_neighbors) = 20, weight function (weights) = distance
[13] http://scikit-learn.org/

We have used a 5-fold cross-validation on 17k data points (2k MWEs and 15k single) collected from the crowdsourcing annotation task for both approaches. The cross-validation is conducted in a way that there is no target overlap in each split so that our model is forced to learn a delexicalized function that can be applied to all targets where substitution candidates are available, cf. (Szarvas et al., 2013).

As evaluation metrics, precision, recall, and F-score are used for the binary classification setup. For the learning to rank setup, we use P@1, Mean Average Precision (MAP), and Normalized Discounted Cumulative Gain (NDCG).

P@1 measures the percentage of correct paraphrases at rank 1, thus gives the percentage of how often the best-ranked paraphrase is judged as correct. A MAP provides a single-figure measure of quality across recall levels.

Normalized Discounted Cumulative Gain (NDCG) (Järvelin and Kekäläinen, 2002; Wang et al., 2013) is a family of ranking measures such as mean average precision (MAP) and Precision at K. NDCG is well-suited for our experiment for its capability of incorporating graded judgments. The graded judgments are obtained from the number of workers selecting a candidate for the given target. NDCG also involves a discount function over the rank while many other measures uniformly weight all positions. NDCG is a normalization of the Discounted Cumulative Gain (DCG) measure. DCG is a weighted sum of the degree of relevancy of the ranked items. The weight is a decreasing function of the rank (position) of the object (discount). NDCG normalizes DCG by the Ideal or maximum DCG (IDCG), which is simply the DCG measure of the best-ranking result. Thus, NDCG measure is always a number in between 0 and 1 where 1 is assigned for a perfect ranking (Wang et al., 2013).

NDCG is computed as follows. Let Cumulative Gain (CG) be the sum of the relevance scores of the candidate paraphrases (y_i) generated for the top k positions (here 5 candidates). CG is is computed as a summation (see Equation 6.1).

$$CG_k = \sum_{i=1}^{k} y_i \tag{6.1}$$

The DCG value is computed using the formula in Equation 6.2.

$$DCG_k = \sum_{i=1}^{k} \frac{2^{y_i} - 1}{log(i+1)} \tag{6.2}$$

Finally the NDCG is computed as shown in Equation 6.3.

$$NDCG_k = \frac{DCG_k}{IDCG_k} \tag{6.3}$$

Here the ideal DCG (IDCG) is the DCG value based on the ideal or real ranking scores of the candidate paraphrases, which are obtained from the suggestions (preferred candidate selection) of the crowd workers.

6.5.1 Binary Classification

For paraphrase selection, we regard the problem as a binary classification task. If a given candidate is selected by at least one annotator, it is considered as a possible substitute and taken as a positive example. Otherwise, it will be considered as a negative training example. For this experiment, the kNN algorithm from the Scikit-learn machine learning framework is employed. Table 6.3.2 shows the evaluation results for the best subsets of feature combinations. The classification experiments obtain a maximal F_1 score of 81.56% for MWEs and 79.77% for single words compared to a non-contextual baseline of 69.06% and 71.47% respectively.

6.5.2 Learning to Rank

We build a learning to rank model to rank candidate paraphrases, using the number of annotators choosing the same candidate as a relevance score (in the interval of $[0–5]$). The average evaluation result on the 5-fold splits is displayed in Table 6.3.2. The baseline ranking based on F_0 only is consistently lower than our context-aware classifiers. The best scores are attained with all features enabled (P@1=89.72, NDCG@5=88.82, and MAP=91.58 for single words vs. P@1=84.69, NDCG@5=77.54 and MAP=86.21 for MWEs). A more detailed analysis between the ranking of single-worded targets and multi-worded paraphrases will be discussed in Section 6.6.3.

	#0	#1	#2	#3	#4	#5	Agreement
All	36.09	34.57	11.68	8.38	5.82	3.46	81.56
Single	36.54	34.47	11.48	8.24	5.79	3.48	81.76
MWE	32.39	35.43	13.35	9.47	6.06	3.30	76.97

Table 6.6.1: Score distributions and observed annotation agreement (in %). The columns #1 to #5 show the percentage of scores the annotators selected to each relevance score (0–5). The last column provides the observed agreements among 5 annotators.

6.6 Analysis of the Result

In this section, we interpret the results obtained during the crowdsourcing annotation task and machine learning experimentation.

6.6.1 Correlation with PPDB2 Ranking

As it can be seen from Table 6.3.1, without considering contexts, a Spearman correlation of 0.36 and 0.25 is achieved by the workers against the PPDB2 default rankings for single and MWE annotations respectively. However, when the contexts are provided to the workers, the ranking for the same items is lower with a Spearman correlation of 0.32 and 0.23 for single and MWE annotations respectively.

From the results (see Table 6.3.1), we can see that the contexts provided have impacts on the ranking of paraphrases. When contexts are not presented, the human judgments and the PPDB2 rankings have a higher correlation. But, when the context is presented, the correlation decreases, which means that the order of candidates selected by the workers is different from the PPDB2 rankings. In general, the correlation score is very low. The reason is, based on our error analysis, there are a lot of inconsistent scores within the PPDB2. For example, the word pairs (*come in*, *sound*) and (*look at*, *okay*) have a high correlation score (3.2, 3.18 respectively). However, they do not seem to be related and are not considered as substitutable by our method. The observed inconsistency is worse in the case of MWE scores hence the correlation is lower than for single words.

6.6.2 Annotation Agreement

As we can see from Table 6.6.1, annotators agree more often on single words than on MWEs. This might be associated with the fact that single word candidates are generated with different meanings using the automatic sense induction approach, provided by the JoBimText framework (Ruppert et al., 2015). Hence, when a context is presented, it is much easier to discern the correct candidate paraphrase. On the other hand, in MWEs, their parts disambiguate each other to some extent, so there are fewer candidates with context mismatches. We can observe that (from the individual class percentages), as MWE candidates are on average scored higher than single word candidates, especially in the range of [2-4], and from the overall observed agreements.

6.6.3 Machine Learning Results

According to the results shown in Table 6.3.2, we achieve higher scores for the binary classification of MWE compared to single words. We found that this is because we have more positive examples (67.6%) for MWE than for single words. Intuitively, it is much easier to have one of the five candidates to be a correct paraphrase as most of the MWEs are not ambiguous in meaning (see recall (R) column in Table 6.3.2).

Example 1: *this is the reason too that the reader disregards the duke 's* ***point of view*** *, and supports and sympathises with the duchess , acknowledging her innocence.*
Example 2: *this list of verbs describes day-to-day occupations of the* ***young girl*** *, suggesting that she does n't distinguish the graveyard from other locations of her day .*
Example 3: *this is apparent in the case of the priest who tries to vanquish* ***the devil*** *, who is infact mistaken for mouse slayer , the cat ...*

Error analysis of the classification result shows that some of the errors are due to annotation mistakes. In Example **1**, the annotators do not select the candidate **stand** while the classifier predicts it correctly. We also found that the classifier wrongly picks antonyms from candidates. The classifier selected **younger man** and **heaven** for Example **2** and **3** respectively while the annotators do not select them. Out of 91 MWE examples predicted by the classifier as positive, 24 have a near synonym meaning that annotators fail to select while, 7 examples are antonyms.

Target	Candidate	#Annotators	Ranker score
write about	write on	2	8.14
write about	write into	0	5.63
write about	discuss	1	2.81
write about	write in	1	1.20
write about	talk to	1	-1.82

Table 6.6.2: Comparison of the Annotators and LambdaMART ranker scores for the phrase **write about** and the different candidates

Once again, we can see that it is difficult to rank candidates appropriately when the candidates provided (in the case of MWEs) are less ambiguous. This could also be a consequence of the lower agreement on MWE candidate judgments. Analysis of the learning to rank result also revealed that the lower result is because more often, the annotators do not agree on a single candidate, as it can be seen from Table 6.6.2.

By looking at the overall results, it becomes clear that our learning framework can substantially improve contextual paraphrase ranking over the PPDB2-resource-based baseline. The resource-based *F*0-feature, however, is still crucial for attaining the highest scores. While the global context features based on word embeddings (cf. $F1 + 2 + 3$ or $F1 + 3$) already show excellent performance, they are consistently improved by adding one or all feature that exhibits local context ($F5, F6, F7, F8$). From this, we conclude that all feature types (resource, global context, local context) are essential to improve the learning to rank performance.

6.7 Conclusion

In this chapter, we have discussed how to build and collect paraphrase resources and quantified the impact of context on the paraphrase ranking and scoring task. The annotation experiments show that paraphrasing is, in fact, a context-specific task: while the paraphrase ranking scores provided by PPDB2 were confirmed by a weak correlation with out-of-context judgments, the correlation between resource-provided rankings and judgments in context were consistently lower.

We have conducted a classification experiment in a delexicalized setting, i.e., training and testing on disjoint sets of targets. For a binary classification as well as for learning to rank setups,

we improved substantially over the non-contextualized baseline as provided by PPDB2. An F-score of 81.56% and 79.87% is achieved for MWEs and single words using the kNN classifier from Scikit-learn for the binary classification setup. A MAP score of 87.14% and 91.58% is obtained for MWEs and single words using the LambdaMART learning to rank algorithm from RankLib.

We conclude that using a learning to rank framework for utilizing features that characterize the candidate paraphrase not only with respect to the target but also with respect to the context is a way forward. The most successful features in these experiments are constructed from word embeddings, and the best performance is attained in a combination of resource-based, global context, and local context features.

Both experiments confirm the generally accepted intuition that paraphrasing, just like the lexical substitution of single words, depends on context: while MWEs are less ambiguous than single words, it still does not hold that they can be replaced without taking the context into account. The collected paraphrase resources are openly available[14], which are distributed under CC-BY license.

In Chapter 9, we discuss the impact of using this generic paraphrase resource to build a baseline paraphrasing model for a specific semantic NLP application, particularly for an adaptive and personalized text simplification task.

[14] https://www.inf.uni-hamburg.de/en/inst/ab/lt/resources/data/learn-multi-word-paraphrasing.html

It's so much easier to suggest solutions when you don't know too much about the problem.

Malcolm Forbes

7

Complex Word Identification Component for Adaptive Text Simplification

In Chapter 6, we have presented how to collect contextualized paraphrase resources that are used to build context-sensitive paraphrasing models. The targets to be paraphrased were determined through a heuristic approach (for example randomly selecting 5 to 8 content words or phrases). If we deal however with a specific NLP application such as text simplification, paraphrasing random target words in the text will not serve its purpose. Instead, it is necessary first to identify the words or phrases that could pose difficulties for a target reader to understand the text.

As parts of the development of adaptive technologies for semantic-aware NLP applications, such as text simplification, we have to start with the identification of complex units in textual documents.

In this chapter, we focus on the methods and processes of complex word identification (CWI),

the collection of CWI datasets, and conduct different experiments to evaluate the quality of the CWI datasets. Also, we report on a shared task that we have co-organized to this end. In Chapter 9, we present a use-case of a semantic-aware NLP application, namely a semantic writing aid for adaptive text simplification that incorporates an adaptive paraphrase ranking component.

7.1 Introduction

Lexically and semantically complex words and phrases can make text understanding difficult for many people, e.g. non-native speakers (Aluísio et al., 2008; Petersen and Ostendorf, 2007), children (De Belder and Moens, 2010), and people with various cognitive or reading impairments (Feng et al., 2009; Rello et al., 2013; Saggion et al., 2015). It has been shown that, for example, people with dyslexia read faster and understand texts better if short and frequent words are used (Rello et al., 2013), while non-native English speakers need to be familiar with about 95% of text vocabulary for a basic text comprehension (Nation, 2001), and even 98% of text vocabulary for enjoying (unsimplified) leisure texts (Hirsh and Nation, 1992).

So far, many guidelines have been published about how to write texts, which are easy-to-understand for various target populations, e.g. Freyhoff et al. (1998); Mencap (2002); Plain-Language (2011), but a manual production of texts from scratch for each target population separately cannot keep up with the amount of information that should be available to everyone on an everyday basis. Therefore, many systems for automatic lexical simplification (LS) of texts have been proposed, mostly for English.

While all LS systems contain modules dedicated to searching for synonyms of potentially complex words, and for ranking the retrieved synonyms by their simplicity and goodness-to-fit in a given context, some of the proposed systems treat all content words in a text as potentially difficult words, e.g. (Glavaš and Štajner, 2015; Horn et al., 2014; **Yimam** et al., 2016b). Meanwhile, other systems have a complex word identification (CWI) module at the beginning of their pipeline, which selects potentially complex words and applies the other two modules only to them, e.g. (Paetzold and Specia, 2016a). The second strategy seems to significantly improve the final results (Paetzold and Specia, 2015).

As the primary purpose of lexical simplification systems is to improve social inclusion of various target populations by offering them more understandable and up-to-date information, most

LS systems focused on simplifying newswire texts, e.g., (Aluísio et al., 2008; Carroll et al., 1999; Glavaš and Štajner, 2015; Saggion et al., 2015). However, given that there are no large parallel corpora of original newswire texts and their manual simplifications, which would enable development of supervised LS systems, with the appearance of the sentence-aligned parallel English Wikipedia – Simple English Wikipedia (EW–SEW) dataset (Coster and Kauchak, 2011), the focus of LS shifted to Wikipedia articles.

This tendency, of course, had an impact on research in CWI as well, leading to only having CWI datasets, which cover the Wikipedia genre (Horn et al., 2014; Paetzold and Specia, 2016c; Shardlow, 2013).

7.2 General and specific objectives

In this chapter, we mainly focus on the collection of gold-standard CWI datasets for English, German and Spanish. We then develop a machine learning system and evaluate the performance of the model. The machine learning models can be used to instantiate a baseline model (see Chapter 9) for an adaptive CWI component, specifically for a semantic-aware text simplification system. Specific objectives include the following:

Objective 1: Collect complex phrases for English covering three genres. Using MTurk, we collect complex phrases both from native and non-native speakers.

Objective 2: Collect complex phrases for German and Spanish. The same design approach used for English is applied to collect complex phrases for German and Spanish. By following the same procedure, we can easily compare results and draw conclusions.

Objective 3: Design and implement multilingual features for a complex phrase identification system. The performance of a supervised machine learning system mainly depends on the preparation and selections of useful features. Moreover, we like to develop language-independent features. The advantage of such a system is that it can easily be extended to a new language with less effort.

Objective 4: Develop machine learning systems for each language and genre and evaluate their performance. Once suitable features are selected, we have to develop the machine learning model and evaluate the performance of the system for each dataset's categories.

Objective 5: Develop and evaluate machine learning systems for different categories such as

cross-genre, cross-group (between native and non native), and cross-language categories.

Objective 6: Report the performance of different systems participating in the CWI shared task 2018, which use the CWI dataset collected, complemented by a new French CWI test dataset collected for multilingual and cross-lingual CWI experiments.

7.3 Related Work

7.3.1 English CWI

Currently, the largest and most used CWI dataset is the SemEval-2016 shared task (Paetzold and Specia, 2016c), which consists of 9,200 sentences collected from the older CW dataset created by Shardlow (2013), LexMTurk Corpus (Horn et al., 2014), and Simple Wikipedia (Kauchak, 2013). Those previous datasets relied on Simple Wikipedia and edit histories as a **gold standard** annotation of CWs, even though the use of Simple Wikipedia as a **gold standard** for text simplification has been disputed (Amancio and Specia, 2014; Xu et al., 2015). The SemEval-2016 CWI dataset, in contrast, is a collection of human annotations of CWs. Another improvement over the previous datasets is that all annotators were non-native English speakers, and therefore the two user groups (native and non-native English speakers) were not mixed as in the previous cases.

In the SemEval-2016 CWI dataset, for each given sentence with only **one** marked target word in it, the annotators were asked to determine if it is complex or not. In the training dataset (200 sentences), each target word was annotated by 20 people, while in the test set (9,000 sentences) each target word was annotated only by a single annotator. The goal of the shared task was to predict the complexity of a word for a single non-native speaker based on the annotations of a larger group of non-native speakers. This introduced a strong bias and inconsistencies in the test set since the test sentences were annotated by only one annotator, but not all of them by the same annotator, involving a total of 400 different annotators. This is reflected in very low F-scores obtained across all systems (Paetzold and Specia, 2016c; Wróbel, 2016).

The systems of the SemEval 2016 shared task were ranked based on their F-score (the standard F_1-measure) and the newly introduced G-score (the harmonic mean between accuracy and recall) on the **complex** class only. Whereas there is a reasonable Spearman correlation be-

tween F-score and G-score considering all systems (0.69) of the SemEval-2016 task, we observe a negative correlation both for the 10 best G-scoring systems (-0.34) and for the best F-scoring systems (-0.74).

The best system with respect to the G-score (77.4%), but at the cost of F-score being as low as 24.60%, uses a combination of threshold-based, lexicon-based, and machine learning approaches with minimalistic voting techniques (Paetzold and Specia, 2016c).

The highest scoring system with respect to the F-score (35.30%), which obtained a G-score of 60.80%, uses threshold-based document frequencies on Simple Wikipedia (Wróbel, 2016). Focusing on the standard F_1-score as the primary evaluation measure in our experiments, we replicate this system on a recent Simple Wikipedia dump[1], which is used as a baseline system.

7.3.2 German CWI

As far as we know, there are no prior works for German and Spanish complex word/phrase identification tasks. The work of Suter et al. (2016) follows a rule-based approach to simplify complex German texts. In this work, rules are developed, which include: character, word, sentence, and text level analysis (applying special character removing, compound word and sentence splitting and text parsing). For lexical simplification, they used vocabulary acronyms (derived from Wikipedia) and the difficult word dictionary from Hurraki, which is available online[2]. However, so far, there is no vocabulary list for simplified German.

The work of Klaper et al. (2013) focuses on the development of a parallel corpus using German/simple German documents as a basis for automatic text simplification systems. The parallel corpus is crawled from the web (to build a statistical machine translation system), where the website provides both 'everyday language' (**Alltagssprache**) and 'simple language' (**Leichte Sprache**).

7.3.3 Spanish CWI

LexSiS, a lexical simplification system for Spanish, was developed to automatically simplify Spanish texts without the need of a parallel corpus (Bott et al., 2012). Lexis assumes that all

[1] https://dumps.wikimedia.org/enwiki/latest/, download date: 25/02/2017
[2] http://hurraki.de/

words, which have an entry in the Spanish **OpenThesaurus** (Baeza-Yates et al., 2016) need a substitute candidate (hence they are CW in our context). Furthermore, the system tries to get the most appropriate substitution set for a given target and finds the best substitution candidate within the set. The best candidate in the set is defined as the simplest and appropriate candidate word for the target. Identification and determining the simplicity of a target word is based on word length and word frequency properties.

7.4 Collection of the New CWI Dataset

We collect the annotations of complex words and phrases (longer sequences of words, up to maximum 50 characters), using the MTurk crowdsourcing platform, from multiple native and non-native English speakers (also collecting the information about whether they are native speakers or not) on three different text genres. Similarly, we collect complex phrases for German and Spanish, using the same UI but in the respective languages. The dataset is freely available distributed under CC-BY license[3].

7.4.1 Data Selection

The English dataset to be annotated comprises of texts from professionally written news, Wiki news (amateur written news), and Wikipedia articles (amateur written encyclopedic articles). For the **News** dataset, we used 100 news stories from the EMM NewsBrief[4] compiled by Glavaš and Štajner (2013) for their event-centered simplification task. For the **WikiNews**, we have collected 42 news articles from the Wikipedia news articles.

To resemble the existing CW resources (Kauchak, 2013; Paetzold and Specia, 2016c; Shardlow, 2013), we also gathered 500 sentences from **Wikipedia**, belonging to different categories (politics, economy, science, and so on) to ensure that we do not introduce a topic bias. For German and Spanish, a total of 978 and 1387 sentences were compiled, which were collected from German and Spanish Wikipedia articles.

[3] https://www.inf.uni-hamburg.de/en/inst/ab/lt/resources/data/complex-word-identification-dataset.html

[4] Freely available at: http://takelab.fer.hr/data/evsimplify/

Dataset	All		Native		Non-native		Both
	Sing.	Mult.	Sing.	Mult.	Sing.	Mult.	
NewsBrief	2,373	10,358	2,032	5,981	1,824	2,923	1,860
WikiNews	1,565	5,687	1,253	4,052	1,091	756	896
Wikipedia	1,170	4,464	1,031	2,792	832	979	773
German	1,525	5,878	1,225	1,727	1,306	3,145	1,166
Spanish	3,983	10,297	3,952	10,080	236	12	172

Dataset	All%		Native%		Non-native%		Both%
	Sing.	Mult.	Sing.	Mult.	Sing.	Mult.	
NewsBrief	18.64	81.36	25.36	74.64	38.42	61.58	14.61
WikiNews	21.58	78.42	23.62	76.38	59.07	40.93	12.36
Wikipedia	20.77	79.23	26.97	73.03	45.94	54.06	13.72
German	20.60	79.40	41.50	58.50	29.34	70.66	15.75
Spanish	27.89	72.11	28.16	71.84	95.16	4.84	1.21

Table 7.4.1: Distributions of collected CPs across all annotators (*All*), native and non-native annotators separately, and the number of CPs selected by at least one native and one non-native annotator (*Both*). The column *Sing.* shows the number/percentage of annotations selected by only one annotator while the column *Mult.* shows the number/percentage of annotations selected by at least two annotators.

7.4.2 Data Collection Procedure

To collect the complex phrase datasets, we have used the annotation tool that is described in Chapter 4 (see Figure 4.1.1 and 4.1.2).

For each language, we follow the same procedure except that the instructions and examples are provided in the same language as the dataset (German instructions for German annotations and Spanish instructions for the Spanish annotations). Every annotation task is created as a HIT (Human Intelligence Task), which consists of 5–10 sentences and is completed by 10 workers each. To annotate complex phrases, workers highlight a single or sequence of words (phrases) using their mouse pointer. In order to control the annotation process, we do not allow users to select simple words such as determiners, numbers and stop words[5], and very long phrases (more than 50 characters).

We also incorporate a compulsory question about whether the annotator is a native speaker

[5] https://github.com/6/stopwords-json/

dataset	uni-gram (%)	bi-gram (%)	tri-gram+ (%)	total
NewsBrief	10,631 (83.50)	1,592 (12.50)	508 (3.99)	12,731
WikiNews	6,242 (86.00)	727 (10.02)	289 (3.98)	7,258
Wikipedia	4,776 (84.77)	661 (11.73)	197 (3.50)	5,634
German	6,832 (92.29)	356 (04.81)	215 (2.90)	7,403
Spanish	11,000 (77.03)	1,975 (13.83)	1,305 (9.14)	14,280

Table 7.5.1: Distribution of collected CW annotations across different text genres and languages with CP lengths.

of each language or not, with an explanation that the answer to this question does not influence the payment. To encourage annotators to read the text carefully and only to highlight complex words, we offer a bonus that doubles the original reward if at least half of their selections match selections from other workers. To discourage arbitrarily massive annotations for the sake of obtaining bonus rewards, we limit the number of selections that annotators can highlight to a maximum of 10 annotations. If an annotator cannot find any complex word, we ask them to provide a comment.

Our data collection is different in several regards from previous works. 1) We allow annotators to select both single words and sequences of words. We think that such datasets are helpful in upstream tasks such as lexical simplification or paraphrasing. 2) We do not show a single sentence at a time, rather multiple sentences (5–10), which allow annotators to select complex phrases considering larger contexts.

7.5 Analysis of Collected Annotations

7.5.1 English CWI Datasets

A total of 181 workers (134 native and 47 non-native) participated in the annotation task, and 25,617 complex phrase (CP) annotations have been collected, out of which 6,830 are unique CPs. Table 7.5.1 shows the distribution across different text genres and CP lengths.

From Table 7.4.1, we see that around 80% of CPs have been selected by at least two annotators. However, when we separate the selections made by native and non-native speakers, we see that: 1) The percentage of selected CPs by multiple native speakers stays stable across different

	Number of Annotators		Avg. annotators per HIT	
dataset	Native	Non-native	Native	Non-native
NewsBrief	67	29	5.8	4.2
WikiNews	56	12	7.6	2.4
Wikipedia	31	13	6.9	3.1
German	12	11	3.9	6.1
Spanish	48	6	9.8	0.2

Table 7.5.2: Distribution of the number of annotators (native and non-native) for each language. The average number of annotators per HIT is computed by averaging the number of total annotators (native and non-native) who have marked at least one complex phrase. For example, in total there are 12 native annotators for German but on average 3.9 annotators visit a HIT and select a complex phrase.

genres, while it is not the case for the non-native speakers. 2) The percentage of CPs selected by many non-native speakers is always significantly lower (54%–62%) than the percentage of CPs selected by multiple native speakers (73%–75%), regardless of the text genre. 3) The percentage of CPs selected by at least one native and one non-native annotator (that is between two groups) is very low (12%–15%).

These results indicate a higher heterogeneity of complex phrases among non-native speakers, raising doubts in how well can we predict complex phrases for a non-native speaker based on the annotations of other non-native speaker, this offers a possible explanation for the very low F-scores obtained by the best systems on the SemEval 2016 shared task. The low inter annotator agreement (IAA) between native and non-native speakers (column *Both*) further indicates that the lexical simplification demands are different for those two target groups.

7.5.2 German CWI Datasets

We have observed that the German CWI annotations collected are different from the English annotations. In general, there are fewer annotators (23 in total, 12 native and 11 non-native) generating 7,403 complex phrases (2,952 for native and 4,451 for non-native annotators) out of which 2,711 are unique CPs. Regarding the number of annotators per HIT, there are more non-native annotators than native annotators (6.1 non-natives and 3.9 natives, see Table 7.5.2).

It is also shown that more than 92% of the annotations are single words in contrast to English

and Spanish. Unlike the English annotators, there is higher IAA among non-native German annotators (70.66%) than native annotators (58.5). This is because there are more non-native annotators per HIT than native annotators. However, we have higher IAA among native and non-native annotators (15.75%) than the English annotators.

7.5.3 Spanish CWI Datasets

Unfortunately, there are very few annotations from non-native Spanish speakers compared to the English and German annotation tasks. There are in total 54 annotators, amongst them, 48 annotators are native speakers, and 6 are non-natives. A total of 14,280 annotations are collected (14,032 from native and 248 non-native annotators) out of which 6,061 are unique CPs.

Unlike the English and German datasets, there is lower IAA for Spanish. This lower IAA for Spanish can be attributed to the fact that annotators highlight largely multi-word expressions or phrases compared to the single words (it equates to around 23% of the annotations, see Table 7.5.1). As the number of annotations from non-native speakers are very few, we do not experiment with the non-native annotations.

7.6 Classification Experiments

We have developed a binary classification system for the CWI task, a performance comparable to the state-of-the-art systems of the SemEval-2016 shared task. We base our discussions on the F-scores, but also report the G-score (both calculated on the *complex* class only, as in the shared task) to compare our systems with the best systems of SemEval-2016.

7.6.1 Machine Learning Features

Four different sets of features are used to build the classification models.

Frequency and length features: Due to the common use of these features in selecting the most simple lexical substitution candidate (Bott et al., 2012; Glavaš and Štajner, 2015), we use three length features (the number of vowels, syllables, and characters) and three frequency features: the frequency of the word in Simple Wikipedia (for German and Spanish, we use the regular

Wikipedia as there are no Simple German Wikipedia and Simple Spanish Wikipedia), the frequency of the word in the paragraph (of a given HIT), and the frequency of the word in the Google Web 1T 5-Grams. Instead of using the raw count of vowels and syllables as a feature, we normalize the count by dividing to the token length. The number of syllables in the word are computed using the *texhyphj*[6] tool, which is a Java implementation of the Liang (1983) hyphenation algorithm and includes implementations for multiple languages.

Syntactic features: The work of Davoodi and Kosseim (2016) indicates that the part of speech (POS) tag influences the complexity of the word. We used POS tags predicted by the Stanford POS tagger (Toutanova et al., 2003).

Word embedding features: Following the work of Glavaš and Štajner (2015) and Paetzold and Specia (2016a), we train a word2vec model (Mikolov et al., 2013) using English Wikipedia and the AQUAINT corpus of English news text (Graff, 2002) for our English datasets. For German and Spanish, we have used the German Wikipedia and Spanish Wikipedia. We train the embeddings with 200 dimensions using skip-gram training and a window size of 5. We use the word2vec representations of complex words as a feature, and also compute the cosine similarities between the vector representations of the word and the paragraph or sentence, which contains it. The paragraph and sentence representations are computed by averaging the vector representations of the content words.

Topic Features: We use topic features that are extracted based on an LDA (Blei et al., 2003) model that was trained based on the English, German, and Spanish Wikipedia articles using 100 topics. The first feature is based on the topic distribution of the word. The second feature captures the topic-relatedness for a word within its context. For this, we compute the cosine similarity between the word-topic vector and the sentence vector, paragraph vector, and document vector representations.

7.6.2 Multilingual Features

Almost all of the features explained in Section 7.6.1 are either language independent (length and frequencies) or can easily be transformed into language-independent features (POS tags, embeddings, and topic features). For the experiments performed across different languages, we

[6] https://github.com/dtolpin/texhyphj

Dataset	POS	Length	Word freq.	Web1T	Embed	Vowel	Syllable
News	42.69	58.66	47.43	57.14	66.43	46.19	57.91
Wiki news	35.68	56.85	48.53	60.05	63.60	48.73	61.63
Wikipedia	36.63	57.28	37.96	67.48	61.28	51.68	60.41
German	47.45	58.38	44.68	51.26	57.20	38.61	47.71
Spanish	51.90	40.64	30.73	17.21	45.39	45.68	43.71

Table 7.6.1: Experimental results on different features. The results under *Word freq.* here are based on the frequencies of the word in the paragraph.

have normalized features to a common scale across languages.

length and frequency features: All the length and frequency related features are normalized as follows: Length of the word is normalized by dividing the length to the average length of all words in the specific language. We have found that, for the English dataset, the average length of a word is 5.3 while for German and Spanish it is 6.5 and 6.2 respectively. Similarly, the frequency of a word in Wikipedia and Web1T corpus is normalized by dividing its frequency to the maximum frequency of the word in the respective corpus.

Syntactic features: POS tags across different languages are very different. For example, for English we can use the Stanford POS tagger (**Penn Treebank**)[7], for German we can use the **Stuttgart-Tübingen tag set (STTS)**[8], and for Spanish we can use the **DEFT Spanish Treebank tag set**[9]. We have transformed the tag sets into a universal POS tag-set based on the work of Petrov et al. (2012)[10]

Multilingual embeddings: The work of Ammar et al. (2016) introduced a single shared embedding space for more than fifty languages. For our task, we have used the trained embeddings model for 12 languages[11].

Topic Features: For our multilingual setup, we use the topic-relatedness feature, which is the cosine similarity between the word-topic vector and the document vector. The work of Boyd-

[7] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

[8] http://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/TagSets/stts-table.html

[9] https://web.archive.org/web/20160325024315/http://nlp.lsi.upc.edu/freeling/doc/tagsets/tagset-es.html

[10] https://github.com/slavpetrov/universal-pos-tags

[11] [http://128.2.220.95/multilingual/data/]

Dataset	Native				Non-Native			
	Train		Test		Train		Test	
	Simple	Complex	Simple	Complex	Simple	Complex	Simple	Complex
NewsBrief	970	459	768	360	1,068	361	860	270
Wiki news	898	531	436	250	1,119	310	516	170
Wikipedia	856	573	268	225	985	444	355	133
German	1117	393	586	187	1014	497	536	238
Spanish	1529	647	1189	435	–	–	–	–
Shared-Task-17	–	–	–	–	1,531	706	84,090	4,131

Table 7.6.2: Distribution of *complex* and *simple* instances in our nine new datasets - in raw counts.

Graber and Blei (2009) suggests to build a multilingual topic model for unaligned text, however, we do not build this model in our work.

7.6.3 Classification Algorithms

We use different machine learning algorithms from the Scikit-learn machine learning framework[12] namely: k-nearest neighbors classifier (KNN), nearest centroid classifier (NC), extra-trees classifier (EXT), random forest classifier (RF), gradient boosting for classification (GB), and support vector machines (SVM). We report only the results of the best classifiers based on **NearestCentroid** (NC)[13].

7.7 Experimental Setups

We first build nine new datasets, six for English (three different text genres each with two different groups of annotators), two for German (native and non-native) and one for Spanish(only native). If at least one annotator marked a word as **complex**, it is considered a positive example. As the SemEval-2016 task has shown that having only one annotator per instance significantly decreases the performance of the system (Wróbel, 2016), we ensure that each instance was attended by at least two annotators (it can be up to ten annotators). In the following sub-sections,

[12] http://scikit-learn.org/stable/supervised_learning.html

[13] The NC is a family of supervised nearest neighbors classifiers that predict the majority label from a predefined number of the closest training samples.

Dataset	Native				Non-Native			
	Train		Test		Train		Test	
	Simple	Complex	Simple	Complex	Simple	Complex	Simple	Complex
NewsBrief	67.88	32.12	68.09	31.91	74.74	25.26	76.11	23.89
Wiki news	62.84	37.16	63.56	36.44	78.31	21.69	75.22	24.78
Wikipedia	59.90	40.10	54.36	45.64	68.93	31.07	72.75	27.25
German	73.97	26.03	75.81	24.19	67.11	32.89	63.73	28.06
Spanish	70.27	29.73	73.21	26.79	–	–	–	–
Shared-17	–	–	–	–	68.44	31.56	95.32	4.68

Table 7.6.3: Distribution of *complex* and *simple* instances in our nine new datasets - in percentage.

we discuss the different setups for our CWI experiments, using the NC classifier and the different groups of features (see Section 7.6.1 and Section 7.6.2).

In all sets of experiments, except the first one, we use training sets of 200 sentences (to produce the same size of training data as in the SemEval-2016 shared task) and the rest of each dataset for testing (controlling for not having the same sentences in training and test sets in all experiments).

The distributions of the *complex* class in our nine new datasets and the SemEval-2016 shared task dataset are presented in Table 7.6.2 and 7.6.3. As it can be noted, the percentages of **complex** instances are similar for both training and test sets in all our datasets, while this is not the case for the SemEval-2016 shared task. The percentage of unbalanced **complex** instances in the training and test sets of the SemEval-2016 shared task is the consequence of the training dataset being annotated by 20 annotators and test set being annotated by only one annotator, and this is probably the cause for the very low F-scores achieved by all systems on the shared task (see Section 7.3). To avoid this problem, we used exactly the same annotation procedure for both training and test sets. In the case of Spanish, there are very few annotations for non-native speakers that are not enough to build a CWI system. Hence, the report for Spanish dataset is only for native annotators.

7.7.1 Setups with Monolingual Features

We have the following setups using the monolingual features described in Section 7.6.1.

- **Setup I:** We run our CWI system on the SemEval 2016 shared task dataset, using the same training/test split as in the original shared task.
- **Setup II:** We run our CWI system on our nine new datasets. We also replicate the best SemEval shared task system based on Simple Wikipedia document frequencies (Wróbel, 2016) for English and Wikipedia document frequencies for German and Spanish. We use document frequencies as a feature for our baseline system.
- **Setup III:** We test whether it is possible to train our CWI system on the annotations of one user group (native or non-native) to predict the CWs for the other user group by comparing the in-group and cross-group system performances on all three text genres of the English dataset and the German dataset. As we do not have enough training and testing set for Spanish non-native, we skip the experiment for Spanish.

7.7.2 Setups with Multilingual Features

The purpose of this setup is to train a model in a source language (either English, German, or Spanish) and test it with another target language (either English, German, or Spanish). Such an experiment will let us know if we can easily train CWI system from a source language to predict CPs in a target language. We have used the set of features explained in Section 7.6.2 for these experiments.

- **Setup IV:** Experiment with multilingual features on the nine datasets. This setup validates if the multilingual features can produce comparable results as the monolingual setups.
- **Setup V:** Train source models in English and test them with German and Spanish. We train models for each English genre using the multilingual features and test the CWI system performance with German and Spanish datasets.
- **Setup VI:** Train source models in German and Spanish and test them with the English genres datasets.
- **Setup VII:** Train source models in German and test the performance of the model with the Spanish dataset and vice versa.

System	G-score	F-score
Our system	**75.51**	**35.44**
Best (G-score) system	77.40	24.60
Best (F-score) system	60.80	35.50

Table 7.8.1: Results on the SemEval-2016 shared task datasets (Setup I)).

7.8 Results and Discussion

Below, we present and discuss the results of each set of experimental setups in the separate subsections.

7.8.1 Monolingual Results

In this section, we discuss experimental results only within a language but comparing results between native and non-native annotators.

7.8.1.1 Results on the Shared Task (Setup I)

The results of our CWI system obtained on the SemEval-2016 shared task dataset, together with the results of the best ranked SemEval-2016 systems by their G-score and F-score, are presented in Table 7.8.1. Our system reaches almost the same F-score as the best F-scored system, but at the same time achieves a significantly better G-score. Similarly, our system reaches almost the same G-score as the best G-scored system, but at the same time achieving a significantly better F-score. The improvement in performance is attributed to the fact that the dataset we have collected is annotated by multiple annotators unlike the shared task, which is annotated only by one annotator.

7.8.1.2 Results on the New Datasets (Setup II)

The results of our NC classifiers and the baseline system and the best F-scored system of the SemEval-2016 shared task (Wróbel, 2016) on our nine new datasets (three genres times two user groups for English, two groups for German and one group of Spanish) are shown in Table 7.8.2 and Table 7.8.3. We obtain similar G-scores on our new datasets as on the shared task

Dataset	F-score		G-score	
	Our (NC)	Baseline	Our (NC)	Baseline
News	**69.22**	66.01	**78.72**	74.79
WikiNews	**69.75**	66.56	**76.78**	75.80
Wikipedia	**67.21**	67.20	70.01	**70.25**
German	**57.03**	51.37	**73.77**	70.74
Spanish	**44.64**	44.04	**59.57**	54.33

Table 7.8.2: Results of our CWI system (NC) and the baseline system on our nine new datasets using the monolingual features (the results of the better of the two systems are in bold) (Setup II - Native datasets).

dataset except for Spanish. However, the F-scores are significantly higher on our datasets than on the shared task dataset. As previously stated, this is probably due to the unbalanced distribution of **complex** words in the shared task training and test sets (Table 7.6.2) (200 training and 9000 test sentences). It might be also due to the fact that the test set of the shared task was annotated by only one annotator (see Sections 7.3 and 7.7).

On all nine datasets, our system performs better than, or equal to, the baseline on both evaluation measures. For all the datasets, our NC system outperforms the baseline system. Our CWI system performs better on the English datasets than on the German and Spanish ones. Also, for English, the system performs better on the native datasets than on the non-native dataset while the reverse is true for German. One reason could be that we have fewer annotators per HIT for English non-native (2.4–4.2 per HIT) than the native speakers (5.8–6.9) (See Table 7.5.2).

The reverse is also true for German (3.9 annotators per HIT for native datasets and 6.1 annotators per HIT for non-native annotators). Even if we have a high number of annotators per HIT as for the Spanish native dataset, the CWI system still performs worse than the others. We suspect that, this lower performance is due to the presence of more sequences of words (multi-words) annotated as complex phrases than for German and English and the CWI system is fitting well to single words (Table 7.5.1 and lower IAA agreement between annotators see Table 7.4.1).

Dataset	F-score		G-score	
	Our (NC)	Baseline	Our (NC)	Baseline
News	**62.35**	60.28	**78.55**	77.42
WikiNews	**57.61**	51.50	72.59	**72.97**
Wikipedia	**57.60**	53.53	**73.41**	69.93
German	**59.25**	56.57	**70.93**	68.99

Table 7.8.3: Results of our CWI system (NC) and the baseline system on our nine new datasets using the monolingual features (the results of the better of the two systems are in bold) (Setup II-Non-native datasets).

7.8.1.3 Results on Cross-Group (Setup III)

The results of the cross-group experiments are presented in Table 7.8.4. When training our CWI system on the English datasets annotated by the native speakers, we have higher F-scores when testing on the datasets annotated by the same group (native speakers) (for example 69.22% for **News** dataset). In the case of training on the English datasets annotated by the non-native speakers, however, the results are the opposite of what we expected, significantly higher F-scores when testing on the datasets annotated by the other group (native speakers) (for example 65.52% for **News** dataset).

When training our CWI system on the German datasets annotated by non-native speakers, we have lower F-Score when testing on the dataset annotated by native speakers (54.51%). However, if we train our CWI system on the datasets annotated by native speakers, we obtain higher F-Score when testing on the datasets annotated by the other groups (non-native speakers). These results imply that the IAA on the test set might impact the results more than the type of the annotator group (Table 7.4.1 shows much higher IAA among native than non-native English speakers, which holds both for the training and test datasets. The reverse is true for the German datasets).

7.8.2 Multilingual Results

In this section, we discuss the results of CWI systems based on multilingual features. As it is described in Section 7.6.2, we have transformed all the features into a language-independent feature space.

Dataset	Training on native		Training on non-native	
	Native	Non-Native	Native	Non-native
News	**69.22**	59.62	**65.52**	62.35
Wiki news	**69.75**	54.95	**67.33**	57.61
Wikipedia	**67.21**	55.46	**67.81**	57.60
German	57.03	**57.89**	54.51	**59.25**

Table 7.8.4: Results of the cross-group experiments: For example, when training on native **News** dataset, the value *69.22* and *59.62* shows the results when the test sets are *Native* and *Non-Native* **news** datasets. (The best results for each training set are in bold) (Setup II).

Dataset	F-score		G-score	
	Our (NC)	Baseline	Our (NC)	Baseline
News	**69.97**	66.01	**79.78**	74.79
WikiNews	**69.25**	66.56	**76.24**	75.80
Wikipedia	**70.79**	67.20	**72.99**	70.25
German	**54.92**	51.37	**73.14**	70.74
Spanish	**45.04**	44.04	**60.72**	54.33

Table 7.8.5: Results of our CWI system (NC) and the baseline system on our nine new datasets using the multi-lingual features (the results of the better of the two systems are in bold) (Setup IV - Native datasets)

7.8.2.1 Baseline result on Multilingual Model (Setup IV)

Table 7.8.5 and 7.8.6 presents the baseline results on different CWI systems developed based on multilingual features. We can see that the baseline and our CWI system results for our nine datasets in Table 7.8.2 and 7.8.3 are comparable (even sometimes better) to the baseline and our CWI system results in Table 7.8.5 and 7.8.6.

7.8.2.2 Cross-Language – English Vs. German and Spanish (Setup V)

In the cross-language CWI systems, we train the source model in one language and test it with datasets in another language (both native and non-native). Table 7.8.7 and 7.8.8 show the results when the CWI models are trained with the English datasets (both native and non-native annotators separately) and tested on German datasets (annotated by native and non-native an-

Dataset	F-score		G-score	
	Our (NC)	Baseline	Our (NC)	Baseline
News	**62.35**	60.28	**78.55**	77.42
WikiNews	**57.89**	51.50	72.73	**72.97**
Wikipedia	**58.31**	53.53	**72.87**	69.93
German	**57.69**	56.57	**69.25**	68.99

Table 7.8.6: Results of our CWI system (NC) and the baseline system on our nine new datasets using the multi-lingual features (the results of the better of the two systems are in bold) (Setup IV - Non-native datasets)

Training	Testing on German		Testing on Spanish	
	Native	Non-Native	Native	Non-native
News	53.89	**58.32**	45.19	–
Wiki news	54.54	**58.42**	44.48	–
Wikipedia	52.93	**58.64**	45.29	–

Table 7.8.7: Results of the cross-language experiments using the English genres as training (the best results on each test set are in bold) (Setup V - Native English genres as training)

notators separately) and Spanish datasets (only native annotators). We can see that testing German datasets annotated by native users show a little drop of performance (2.5–4% on each genre), but testing on German datasets annotated by non-native speakers and Spanish datasets annotated by native speakers produce similar performance compared to the monolingual results.

Training	Testing on German		Testing on Spanish	
	Native	Non-Native	Native	Non-native
News	53.02	**58.92**	44.79	–
Wiki news	56.03	**58.31**	43.26	–
Wikipedia	51.53	**59.14**	44.39	–

Table 7.8.8: Results of the cross-language experiments using the English genres as training (the best results on each test set are in bold) (Setup V - Non-Native English genres as training)

Training	Testing on news		Testing on Wiki news		Testing on Wikipedia	
	Native	Non-Native	Native	Non-native	Native	Non-native
German native	**67.42**	57.55	**66.79**	57.08	**62.14**	51.22
German non-native	**66.99**	58.51	**64.17**	55.53	**63.78**	54.09
Spanish	**66.05**	56.37	**62.03**	51.89	**62.04**	56.15

Table 7.8.9: Results of the cross-language experiments using the German and Spanish datasets as training (the best results on each test set are in bold)(Setup VI)

Training	Testing on Spanish		Testing on German	
	Native	Non-Native	Native	Non-Native
German native	42.76	–	–	–
German non-native	41.52	–	–	–
Spanish native	–	–	53.53	56.82

Table 7.8.10: Results of the cross-language experiments between German and Spanish datasets (the best results on each test set are in bold) (Setup VII)

7.8.2.3 Cross-Language – German and Spanish Vs. English (Setup VI)

Table 7.8.9 shows cross-language CWI performance results when German datasets annotated by native and non-native speakers and Spanish dataset annotated by native speakers are used as training. It also shows when the three English genres datasets annotated by native and non-native speakers are used to test the performance of the system. We can see that, when the CWI system is trained on the German and Spanish datasets, testing with English datasets still performs very well.

7.8.2.4 Cross-Language – German Vs. Spanish (Setup VII)

The last cross-language experiment is performed using the German datasets annotated by native and non-native speakers to train the CWI system and test with the Spanish dataset annotated by native speakers and vice versa. While there is about 2% reduction in the F-score results, we still consider the result as reasonably good.

7.9 Report and Summary of the CWI Shared task 2018

The multilingual CWI datasets are subsequently used for the CWI Shared task 2018, which is a follow up of the CWI shared task 2016 (Paetzold and Specia, 2016c). In this section, we present 1) how we clean up further the dataset and prepare it for the CWI shared task participants, 2) the extra French dataset that has been collected using a similar approach to expand the multilingual and cross-lingual setup of the shared task, and 3) the approaches, experiments, features, and results of the shared task participants.

7.9.1 Dataset Preparation

The CWI datasets have been further preprocessed. Some of the preprocessing and cleanup operations performed include 1) splitting sentences for each language properly with a better sentence segmenter and manually inspect to see if sentences are distorted, 2) analyzing and cleaning complex phrases, and 3) balancing the number of native and non-native annotators particularly for the English dataset collection.

Sentences that have been wrongly split or merged have been extensively inspected. Similarly, some of the complex phrases either include non-word characters (such as spaces, quotes, commas, brackets, named entities, and so on), or they are excessively longer than expected. Table 7.9.1 presents the preprocessed datasets distributed for the shared task participants.

Language	Train	Dev	Test
English	27,299	3,328	4,252
German	6,151	795	959
Spanish	13,750	1,622	2,233
French	-	-	2,251

Table 7.9.1: The number of instances for each training, development, and test set

7.9.2 The French Dataset Collection

The French dataset has been collected using the same method as the CWI datasets described in the previous sections (for more details, see our papers **Yimam** et al. (2017b,c)). The dataset

contains Wikipedia texts extracted from a comparable simplified corpus collected by Brouwers et al. (2014). For each article, all paragraphs containing between 5 and 10 sentences are extracted. From this pool of paragraphs, only the best paragraph is selected via a ranking procedure maximizing sentence length and lexical richness and minimizing the ratio of named entities and foreign words. An optimal subset of 100 paragraphs is then selected using a greedy search procedure similar to that of Tack et al. (2016), minimizing the vocabulary overlap between pairs of paragraphs using the Jaccard coefficient, of which, a random test split of 24 paragraphs is selected to be annotated.

7.9.3 Task Setups

The goal of the CWI shared task of 2018 is to predict which words will challenge non-native speakers based on the annotations collected from both native and non-native speakers. The participants are provided with the labeled CWI datasets for each of the monolingual datasets except for the French dataset, which is provided only for the multilingual CWI track for test result submissions.

Given the CWI datasets prepared, the CWI challenge was divided into four different tracks as a) **English monolingual CWI**, b) **German monolingual CWI**, c) **Spanish monolingual CWI**, and d) **Multilingual CWI with a French test set**

Unlike the CWI 2016 shared task, participants are provided with the labeled CWI datasets intended for binary classification (complex or simple) or graded label (from very simple, represented by the minimum value 0.0, to very complex, represented by the maximum value 1.0).

- **Binary classification task:** Participants were asked to label the target words in-context as complex (1) or simple (0).
- **Probabilistic classification task:** Participants were asked to assign the probability of a target word being complex given the context.

Participants were free to choose the task/track combinations they would like to participate.

7.9.4 Baseline Systems

For both the binary and probabilistic classification tasks, we build a simple baseline system that uses only the most basic features described in Section 7.7.1 and 7.7.2, namely only frequency and length features. The Nearest Centroid classifier and the Linear Regression algorithms from the scikit-learn machine learning library are used for the binary and probabilistic classification tasks respectively. For the binary classification task, we have used the accuracy and macro-averaged F1 evaluation metrics. For the probabilistic classification task, the Mean Absolute Error (MAE) measure is used. The baseline results are shown in Table 7.9.2, 7.9.3, and 7.9.4 for the monolingual and multilingual tracks.

7.9.5 Shared Task Systems

A total of 12 teams participated, submitting 252 runs of their systems. The majority of the systems are based on ensemble methods (Random Trees, Random Forests, Extra Trees, AdaBoost, XGBoost) while neural network approaches including Feedforward neural network (FFNN), long short-term memory (LSTM), and convolutional neural network (CNN) are used by many systems compared to the CWI shared 2016 systems.

Features: While 85% of the systems have used **word length** as a feature, more than 50% of the systems have used n-grams, word embeddings, psycholinguistic properties (such as familiarity, concreteness, age of acquisition, and imagery) (Paetzold and Specia, 2016b), word frequency, semantic features (WordNet), and word length as features.

7.9.6 System Results

In this sub-section, we present the results of the participating teams for monolingual and multilingual classification and probabilistic classification tasks.

Most systems follow the traditional feature engineering-based approaches (mainly based on length and frequency features), whenever they perform better than neural network and word embedding-based approaches. However, compared to the SemEval 2016 task results, we have observed that more systems employed deep learning approaches and the results are getting better for the CWI task; the difference is less pronounced for the probabilistic classification tasks.

	team	paper	avg. rank	News	Wikinews	Wikipedia
1	CAMB	Gooding and Kochmar (2018)	1	0.8736	0.8400	0.8115
2	NILC	Hartmann and dos Santos (2018)	3	0.8636	0.8277	0.7965
3	ITEC	De Hertog and Tack (2018)	4.33	0.8643	0.8110	0.7815
4	NLP-CIC	Aroyehun et al. (2018)	4.67	0.8551	0.8308	0.7722
5	CFILT	Wani et al. (2018)	5.33	0.8478	0.8161	0.7757
6	UnibucKernel	Butnaru and Ionescu (2018)	6	0.8178	0.8127	0.7919
7	SB@GU	Alfter and Pilán (2018)	6	0.8325	0.8031	0.7832
8	TMU	Kajiwara and Komachi (2018)	6.33	0.8632	0.7873	0.7619
9	hu-berlin	Popović (2018)	9	0.8263	0.7656	0.7445
10	LaSTUS/TALN	AbuRa'ed and Saggion (2018)	10.33	0.8103	0.7491	0.7402
	Baseline			0.7579	0.7106	0.7179
			rank	Spanish		
1	TMU	Kajiwara and Komachi (2018)	1	0.7699	-	-
2	ITEC	De Hertog and Tack (2018)	2	0.7637	-	-
3	NLP-CIC	Aroyehun et al. (2018)	3	0.7672	-	-
4	CoastalCPH	Bingel and Bjerva (2018)	4	0.7458	-	-
5	SB@GU	Alfter and Pilán (2018)	5	0.7281	-	-
6	hu-berlin	Popović (2018)	6	0.7080	-	-
	Baseline			0.7237	-	-
			rank	German		
1	TMU	Kajiwara and Komachi (2018)	1	0.7451	-	-
2	SB@GU	Alfter and Pilán (2018)	2	0.7427	-	-
3	hu-berlin	Popović (2018)	3	0.6929	-	-
4	CoastalCPH	Bingel and Bjerva (2018)	4	0.6619	-	-
	Baseline			0.7546	-	-

Table 7.9.2: Binary classification results (F1) for the monolingual tasks.

	team	paper	avg rank	News	Wikinews	Wikipedia
1	CAMB	Gooding and Kochmar (2018)	1.33	0.0558	0.0674	0.0739
2	ITEC	De Hertog and Tack (2018)	2.33	0.0539	0.0707	0.0809
3	TMU	Kajiwara and Komachi (2018)	2.33	0.0510	0.0704	0.0931
4	NILC	Hartmann and dos Santos (2018)	3.66	0.0588	0.0733	0.0819
5	SB@GU	Alfter and Pilán (2018)	5	0.1526	0.1651	0.1758
	Baseline			0.1127	0.1053	0.1112
			rank	Spanish		
1	TMU	Kajiwara and Komachi (2018)	1	0.0718	-	-
2	ITEC	De Hertog and Tack (2018)	2	0.0733	-	-
3	CoastalCPH	Bingel and Bjerva (2018)	3	0.0789	-	-
	Baseline			0.0892	-	-
			rank	German		
1	TMU	Kajiwara and Komachi (2018)	1	0.0610	-	-
2	CoastalCPH	Bingel and Bjerva (2018)	2	0.0747	-	-
	Baseline			0.0816	-	-

Table 7.9.3: Probabilistic classification results (MAE) for the monolingual tasks.

	team	paper	F1	MAE
1	CoastalCPH	Bingel and Bjerva (2018)	0.7595	0.0660
2	TMU	Kajiwara and Komachi (2018)	0.7465	0.0778
3	SB@GU	Alfter and Pilán (2018)	0.6266	-
4	hu-berlin	Popović (2018)	0.5738	-
	Baseline		0.6344	0.0891

Table 7.9.4: Binary and probabilistic classification results for the multilngual tasks (French as test set).

One of our notable findings is that the cross-lingual experimental results are very promising, which we think is significant progress for CWI research. Even though we did not provide a training dataset for French, the results obtained have better or equivalent scores (though they, of course, cannot be directly compared) to the German and Spanish datasets, when the system uses either one or several training datasets from the other languages.

7.10 Conclusions

Complex word identification (CWI) task is an essential task in text accessibility and text simplification. So far, however, this task has only been addressed on the Wikipedia sentences and considering only the needs of non-native English speakers. Also, other languages than English do not receive more attention to CWI experiments.

We address the CWI problem for multiple genres of English texts and multiple languages. First, we follow similar procedures to collect complex phrases for multiple languages using MTurk crowdsourcing platform. In addition to preparing the interface in different languages, namely English, German, Spanish, and French, we also ask if they are native speakers of the target language or not.

To enable the construction of generalizable and more reliable CWI systems, we have collected a new complex phrase identification dataset across different text genres, annotated both by native and non-native speakers.

The analysis of our crowdsourced data showed that native speakers have a higher inter-annotator agreement (IAA) on the task than the non-native speakers regardless of the text genre. We also observe that the native speaker's IAA is stable across the different genres, unlike the IAA within the non-native speakers.

Analysis of the collected complex phrases shows that 1) IAA for both native and non-native speakers are much higher for English and German than Spanish, 2) for English, native annotators have higher IAA, which is also stable across different text genres, 3) non-native annotators for German shows higher IAA than the native annotators. Furthermore, Spanish annotators tend to select more multi-word-expressions than English and German.

We built a CWI system comparable to the state-of-the-art and showed that predicting the complex words selected by native speakers is an easier task compared to predicting the complex

words of non-native speakers. We further showed that in-genre CWI indeed leads to better classification performance but that the results are not much worse even for a cross-genre task.

Additionally, we demonstrated that using larger contexts (5–10 sentences instead of only one sentence) for the extraction of CWI effects in better CWI results.

Furthermore, the datasets have been used to organize the CWI shared task 2018. To further extend the multilingual and cross-lingual CWI experiments, a CWI dataset for French has been collected with the same approach as for the other languages. We have found that multilingual and cross-lingual CWI approaches are viable.

Part IV

Exploring Rapid Annotation, Annotation Automation, and Personalized/Adaptive NLP Applications

In Chapter 3, 4, and 5, we have presented different annotation and automation tools as well as semantic-aware adaptive and personalized NLP applications using an embedded model. In Chapter 6 and Chapter 7, different resources that are used to build machine learning models for semantic technologies have been presented. In the next two chapters, Chapter 8 and Chapter 9, we present the automation strategies and processes, the human-in-the-loop approaches, and the experiments we conducted for the adaptive annotation tools and personalized semantic-aware NLP applications.

Tell me and I forget. Teach me and I remember. Involve me and I learn.

Benjamin Franklin

8

Annotation Automation & Human-in-the-loop – Rapid Dataset Development

In Chapter 2, we have briefly presented the different annotation tasks, the issues related to different annotation problems, and the different types of annotation tools that are used to collect training data for NLP applications that rely on machine learning components. In this chapter, we discuss in detail the experimental setups, results, and the analysis of adaptive annotation approaches using these tools.

We know that annotation tasks, in general, are costly for many reasons. First, it requires experts in the area to prepare a detailed annotation guideline. Second, the annotation process takes too much time to collect the required training datasets. Third, the training datasets collected might be inadequate to serve the expected requirements due to a **concept** or **semantic** drift problems (change of concepts or semantics of words over time due to various factors).

To complete an annotation task faster, we have presented different rapid annotation approaches

in Chapter 3 and Chapter 4. In this regard, the integration of an automatic annotation suggestion model with the capability of correcting the suggestions by annotators is one of the parameters of a rapid annotation process.

A rapid annotation process that incorporates an automation component to produce suggestions and allow annotators to correct the suggestions is widely known as the **human-in-the-loop** annotation paradigm. The most important aspect of the human-in-the-loop annotation paradigm is to employ annotators to correct annotations that are suggested by the automation model, in addition to carrying out the regular annotation task. The corrected annotations further help to update and improve the model, for interactive and iterative annotation setups.

Based on the automation component in WebAnno (cf. Chapter 3), this chapter focuses on conducting annotation automation experiments and analyzing the impact of the rapid annotation process, particularly concerning the annotation time, the annotation quality, and its usability for different annotation problems.

8.1 Automation Integration in WebAnno

The integration of machine learning (the automation component) into WebAnno supports exhaustive annotation of documents providing a shorter loop than standard annotation tools because new documents are added to the training set as soon as they are completed by the annotators. The machine learning functionality in WebAnno is applicable to sequence classification tasks. The Automation component in WebAnno also offers high-quality and easy-to-configure inbuilt automation that constitutes a further step of making WebAnno a general-purpose, one-stop-shop tool for multiple types of annotation projects.

8.2 Related Work

The impact of using lexical and statistical resources to produce annotation automatically for increasing annotation speed has been studied widely for various annotation tasks. For the task of medical named entity labeling, Lingren et al. (2013) investigate the impact of automatic suggestions on annotation speed and potential biases using dictionary-based annotations. This technique results in 13.83% to 21.5% time saving and the inter-annotator agreement (IAA)

increased by several percentage points.

WordFreak (Morton and LaCivita, 2003) includes an automation component for different tasks, such as POS tagging and coreference resolution, where instances with a low machine learning confidence are presented for annotation in an active learning setup. Beck et al. (2013) demonstrate that the use of active learning for machine translation reduces the annotation effort. They were able to demonstrate a reduction in the need for manual annotation on three out of four datasets.

GoldenGATE editor (Sautter et al., 2007) integrates NLP tools and assistance features for manual XML editing. The tool is used in correcting/editing an automatically annotated document with an editor where both text and XML markups are modified. GoldenGATE is merely used to facilitate the correction of an annotation while the pre-annotation is completed outside of the tool.

Automatic annotation support in Brat (Stenetorp et al., 2012) was carried out for the semantic class disambiguation task to investigate how such automation facilitates the annotators' progress. They report that there was a 15% reduction in total annotation time. Unfortunately, the automation process in Brat 1) depends on bulk annotation imports and web service configurations that are labor intensive, 2) is task-specific so that it requires a lot of effort to adapt to different annotation tasks, 3) there is a limited way of re-using the corrected result for the next iteration of training the automatic tool.

The GATE Teamware (Bontcheva et al., 2013) automation component comes closest to our work. It is based on either plugins and externally trained classification models or uses web services. Thus, it is highly task specific and requires extensive configuration. The automatic annotation suggestion component in our tool, in contrast, is easily configurable and adaptable to different annotation tasks, and allows the use of annotations from the current annotation project – probably at the cost of an accuracy reduction (mostly at the beginning of the annotation task) when compared to more tailored components.

8.3 Preliminary Experiments for Annotation Automation

As a preliminary experiment, we have conducted an interactive machine learning simulation to investigate the effectiveness of an adaptive approach for named entity annotation and POS

tagging tasks.

For the named entity annotation task, we have used the training and development dataset from the GermEval 2014 Named Entity Recognition Shared Task (Benikova et al., 2014a) using an online machine learning algorithm called MIRA[1] (Crammer and Singer, 2003). The training dataset is divided by increasing size, as shown in Table 8.8.1 to train the system where every larger partition contains sentences from earlier parts. From Figure 8.3.2 it is evident that the adaptive and interactive machine learning approach improves the performance of the system (increase in recall) as users continue correcting the suggestions provided.

Regarding annotation time gain, we produce annotation suggestions (see Figure 8.3.1) for the test splits based on a model trained on the training and validation datasets. While attaining an F-score of about 0.8, it leads to an increase in annotation speed of about 21% when leveraging automation.

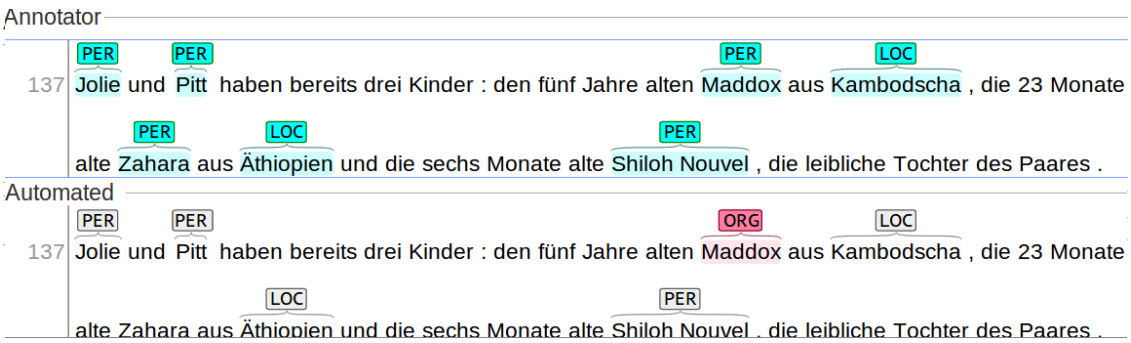


Figure 8.3.1: Sample result of German named entity automation in WebAnno.

Furthermore, in addition to the simulation experiments for the NER task, we have conducted an automation annotation experiment has been carried out for Amharic POS tagging to explore if adaptive and interactive machine learning reduces annotation time. In this experiment, a total of 34 sentences are manually annotated, simulating different levels of precision and recall (refer in Table 8.3.2) for automatic suggestions as shown in Figure 8.3.3. We have conducted this annotation task several times to measure the savings in time when using automatic annotation. When no suggestion was provided, it took about 67 minutes for an expert annotator to annotate the document entirely. In contrast to this, the same annotation task with suggestions (e.g.,

[1] https://code.google.com/p/miralium/

Sentences	Precision	Recall	F-score
24	80.65	1.12	2.21
60	62.08	6.68	12.07
425	71.57	35.13	47.13
696	70.36	43.02	53.40
1264	71.35	47.15	56.78
5685	77.22	56.57	65.30
8770	77.83	60.16	67.86
10812	78.06	62.72	69.55
15460	78.14	64.96	70.95
24000	80.15	68.82	74.05

Table 8.3.1: Evaluation result for the German named entity recognition task using a simulation of an online learning approach with different sizes of the training dataset tested on a fixed development dataset.

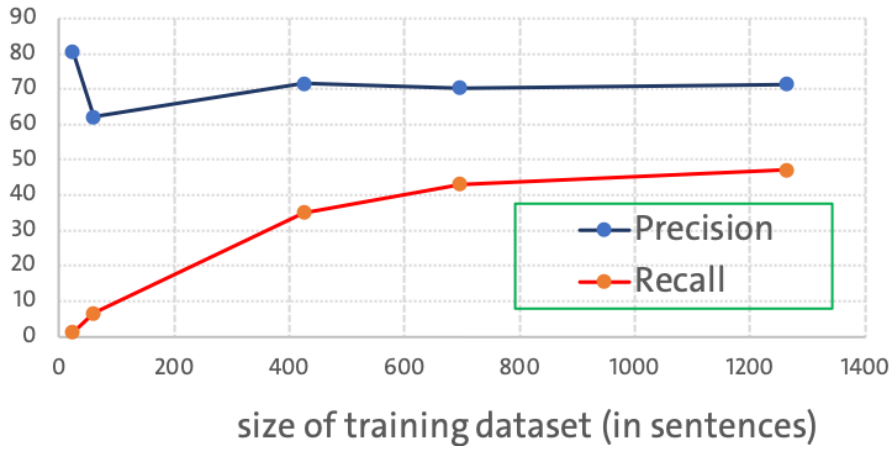


Figure 8.3.2: Learning curve showing the performance of the adaptive and interactive automation using different sizes of the training dataset

with a recall of 70% and precision of 60%) took only 21 minutes, demonstrating a significant reduction in annotation cost.

Furthermore, we have conducted POS tag annotation in WebAnno to build POS tagged datasets for Amharic. Amharic is an under-resourced language in the Semitic family, mainly spoken in Ethiopia. POS tagging research for Amharic is mostly attended as an academic exercise. The latest result reported by Gebre (2009) was about 90% accuracy using the Walta

	no auto.		
no Auto.	67	Recall	
Precision		30	50	70
	60	53	33	21
	70	45	29	20
	80	42	28	18

Table 8.3.2: Experimentation of adaptive machine learning for different precision and recall levels for Amharic POS tagging task. The cell with the precision/recall intersection records the total time (in minutes) required to fully annotate the dataset with the help of adaptive automation. Without automation (no auto.), annotation of all sentences took 67 minutes.

Figure 8.3.3: Amharic POS tagging. lower pane: suggestion provided to the user by the interactive and adaptive classifier, upper pane: annotations by the user. When (grey) the suggestion in the lower pane is correct, the user will click the annotation and copy it to the upper pane. Otherwise (shown in red or no suggestion), the user should provide a new annotation in the upper pane.

Information Center (WIC) corpus of about 210,000 tokens (1065 news documents). We intentionally do not use the corpus as training data because of the reported inconsistencies in the tagging process (Gebre, 2009). Instead, we manually annotate Amharic documents for POS tagging both to test the performance of the automation module and to produce POS-tagged corpora for Amharic.

Tag-set: Based upon the work by Petrov et al. (2012) and Ethiopian Languages Research Center (ELRC) tag-sets, we have designed 11 POS tags equivalent to the Universal POS tag-sets. The tag *DET* is not included as Amharic denotes definiteness as noun suffixes.

Collection of documents and preprocessing: We collected some Amharic documents from an online news portal[2]. Preprocessing of Amharic documents includes the normalization of characters and tokenization (sentence and word boundary detection).

Annotation and automation: We manually annotated an initial of 21 sentences. An iterative automatic annotation suggestion process was started using these sentences as training data until 300 sentences were fully annotated. We obtained an F-score of 0.89 with the final model, which is compared againest manually annotated sentences. Hence the automatic annotation suggestion component helps in decreasing the total annotation time since the user has to manually annotate only one out of ten words while being able to accept most of the automatically induced suggestions.

Figure 8.3.4 shows an example Amharic document processed employing our approach.

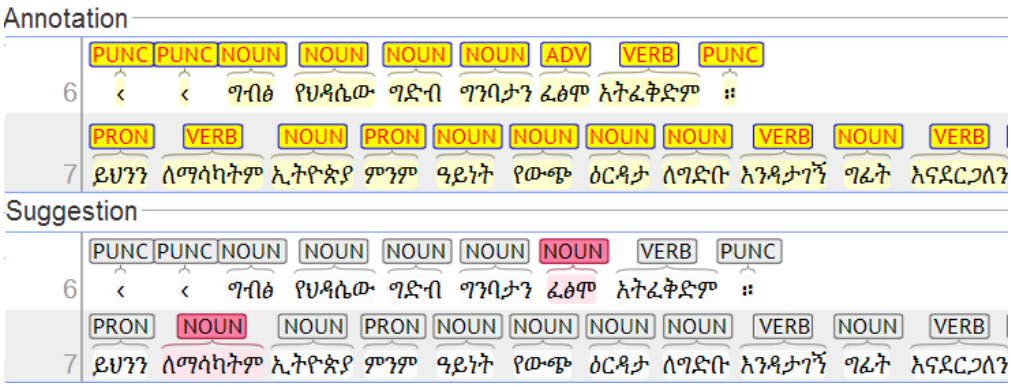


Figure 8.3.4: Automation suggestion example for Amharic document using the 11 Universal POS tag-set. The red tags in the suggestion pane have not been confirmed by the annotator.

8.4 Automation and Human-in-the-loop for Biomedical Entity and Relation Recognition

The biomedical domain is increasingly turning into a data-intensive science, and the challenge concerning the ever-increasing body of medical literature is not only to extract meaningful information from this data, but also to gain knowledge, insight, and to make sense of the data

[2] http://www.ethiopianreporter.com/

(Holzinger, 2013). Texts are an essential type of data within the biomedical domain. As an example in the medical domain, patient records contain large amounts of text, which have been entered in a non-standardized format, consequently posing a lot of challenges to the processing of such data and for the clinical doctor the written text in the medical records is still the basis for any decision making (Holzinger et al., 2008, 2014). Further, scientific results are communicated in text form, consequently for the biomedical domain text is an indispensable data type for gaining knowledge (Holzinger et al., 2013).

Modern automated information extraction (IE) systems usually are based on machine-learning models, which require a large amount of manually annotated data to specify the model according to the task at hand. Unfortunately, particularly in the medical domain, experts have obligations with higher priorities. Thus, it is costly and cumbersome to annotate a large number of training examples. To alleviate this problem, there is a need for an approach where human annotators are facilitated to annotate faster than the traditional way, in order to produce the required annotations in less time.

While we have seen tremendous efforts in the past years to standardize and link lexical taxonomies and ontologies[3], there has not been widespread use of such structured resources for the formal representation of the semantics of text. We attribute this to their excessive size and their author-centricity, as well as to the lack of information for being able to assign their concepts to respective terms in the unstructured text. Just because, e.g., all viruses are known in a database, it does not follow from this that it is possible to find their occurrences in a text (e.g., because of ambiguous abbreviations, short forms and variants, and idiosyncrasies of the subfield).

Here, we propose a radical break with this traditional way of knowledge representation: instead, **users** should be able to choose their own set of categories per given task or problem, and thus should be able to grow their own local ontology without the need (but eventually with the possibility) of connecting it to existing upper ontologies, and users should **ground** their conceptualization in the respective **texts** of their current interest.

We specifically tackle the extractions of entity mentions and their relations from biomedical texts, specifically from MEDLINE abstracts[4], using a human-in-the-loop automation strategy that has not been applied in the medical domain before. Unlike named entity recognition

[3] see http://www.w3.org/wiki/LinkedData

[4] www.ncbi.nlm.nih.gov/pubmed

(NER) systems on, e.g., the news domain, entity recognition on medical domains comprises of extractions of technical terms in the broader medical and biological arena such as the name of diseases, proteins, and substances, see, e.g., (Ghiasvand and Kate, 2014; Leser and Hakenberg, 2005).

Such an automation approach is especially essential for the medical domain, as a full manual annotation is extremely expensive. Medical professionals in turn, however, are willing to perform this task only diligently if it matches their current field of interest. The human-in-the-loop automation approach enables users to start the automation process without pre-existing annotations and works by suggesting annotations as soon as the users have annotated a rather small number of documents. This **annotate-little** and **predict-little** strategy is deemed adequate for biomedical domains as it 1) produces quality annotation in a very short period of time, 2) is adaptive in such a way that newly evolving concepts or entities will not be ignored by an old and static prediction classification model, and 3) the conceptualization (i.e., entity types and their typed relations) can be chosen and extended by the user during the annotation process. Thus, this human-in-the-loop approach follows the principles of the recently emerging cognitive computing paradigm that proposes more adaptive, iterative, and interactive human-machine interaction (Holzinger, 2016).

Note that while models trained on a small number of entities mentions cannot be expected to produce high-quality automatic labels, however, their annotation suggestions might still be useful for the task at hand, in turn, helping to produce more annotations in a short time that eventually improve the quality of the automatic labels.

We conduct three experiments to exemplify and evaluate the adaptive and human-in-the-loop approach of entity mention annotation for the medical domain. In the first experiment (Section 8.8.1), we simulate the interactive machine learning approach by incrementally processing the BioNLP-NLPBA 2004 named entity annotated dataset (GuoDong and Jian, 2004). During the simulation, a classifier model is first trained on very few annotations, and we measure the number and quality of correctly predicted annotations in the next chunk of the data, which subsequently is added to the training, simulating the annotation process. With this simulation, we can learn whether annotating very few documents already produces reasonable and faithful predictions so that it relieves users from annotating every document in the dataset.

In the second experiment, we put our approach to practice and apply it in a use case where

medical professionals annotate documents to support research on their particular question of interest. Specifically, the task used for this study is focused on the investigations of the causes of the **B-chronic lymphocytic leukemia (B-CLL)** on MEDLINE abstracts and users annotate terms with their respective entity classes with **span annotations** (see Chapter 3), which means that annotators assign an entity label to a word or multi-word expressions in the text. Here, we compare two setups where annotators are presented, or not presented with suggestions from the classifier in the adaptive and interactive annotation interface. This experiment sets out to clarify whether medical professionals perceive our human-in-the-loop approach as appropriate and helpful in terms of quantitative and qualitative assessments.

The third experiment extend this notion further: here, we focus on the relations between such entities, which is a more interesting type of knowledge from an application perspective but also poses a more challenging problem for incremental machine learning. In this experiment, we let our medical expert annotate, e.g., the interactions between proteins or relations between antibodies and antigenes. We notice that the system quickly picks up on user-defined relations, and found that our medical expert had to define new relations given in a standard dataset in order to model the requirements.

This experiment specifically target the research problem "**How could a machine learn from usage data?**" demonstrated by the following three achievements: First, we show how using the human-in-the-loop approach, we can outperform an approach that relies only on expert annotation without the human in the loop. Second, we demonstrate that even with a little amount of annotation, a good performance for annotation suggestion can be reached, resulting in a substantial annotation speedup. Third, we exemplify how the human-in-the-loop approach in text annotation allows the customization of entities and relation types for the user's need.

8.5 Literature on Annotation Automation for a Biomedical Domain

The following subsections give a brief overview of related work in adaptive machine learning for named entity tagging and relation learning for the medical domain in general.

8.5.1 Human into the Loop

Automated machine learning algorithms work well in certain environments. However, biomedical data are full of probability, uncertainty, incompleteness, vagueness, and noise that makes the application of automated approaches difficult, yet often impossible. Moreover, the complexity of current machine learning algorithms has discouraged medical professionals from the application of such solutions.

There is also the issue of acceptability and provenance, (see Biemann (2014)): since their decisions might be life-critical, medical professionals will not accept automatic systems, even with high precision, which cannot justify the rationale for the automatic decision. While there exist rather simple learning algorithms (e.g., memory-based learning, (Daelemans et al., 1998)) that provide digestible explanations (e.g., in form of similar examples or situations), they need more training data to reach the same performance level as more advanced and complex algorithms (e.g., deep learning (Goodfellow et al., 2016)).

However, most complex approaches fail to give interpretable reasons for their automatic classification, which calls for facilitation of training data creation, especially for sensitive and life-critical domains; a further drawback with complex machine learning approaches is that their training is done in epochs over the whole dataset and there is no straightforward way to add additional labeled examples to the model.

Hence, for increasing the quality of such approaches, the integration of the expert's domain knowledge is indispensable. The interaction of the domain expert with the data would greatly enhance the whole knowledge discovery process chain. Interactive and adaptive machine learning puts the human into the loop to enable what neither a human nor a computer could do on their own, (see Holzinger (2013)). For this, only machine learning algorithms are suitable that support online learning. We use a perceptron-based online learning algorithm to generate suggestions of manual text annotation for the biomedical entity automation task. These annotations are subsequently used to generate better suggestions, as the model continuously updates based on human interaction with our annotation tool.

8.5.2 Interactive and Adaptive Learning

Static machine learning model assumes that the actual state of the "domain universe" can be sufficiently acquired by listing all available datasets at a particular time. In contrast, adaptive machine learning assumes the possibility that there might exist unrecorded facts at a particular time, which can only appear at some point in the future. This, however, is rather the standard situation than the exception: think for example a recommendation system for an online shopping platform: if it was static, there would be no recommendations for any product that was launched after the system was set up.

Authors of Ludl et al. (2008) address an industrial case study (tile manufacturing process) and found out that the classical machine learning setup faced difficulties such as 1) feedback is usually obtained after a process is completed, which might help the system, 2) some variables can change through time, and 3) error correction is always done after observation. The research by Drucker et al. (2011) on clustering a large number of documents using an interactive recommender system shows that users can sort documents into clusters significantly faster with an interactive recommender system than correcting the output of a static automated method. On top of simple user feedback in (Stumpf et al., 2007), such as accepting and rejecting suggestions, complex feedback like choosing the best features, suggestions for the re-weighting of features, proposing new features and combining features remarkably improve the system.

8.5.3 NER for Medical Domains

Recent years have seen a surge on biomedical text processing (see Cohen and Hersh (2005) for a survey), most of which rely on the GENIA corpus (Ohta et al., 2002), which is a collection of biomedical abstracts. It is mainly annotated for linguistic structures such POS tagging and syntax annotation, semantic annotation of entities and so on (Tateisi and Tsujii, 2004; Tateisi et al., 2005). The work of Lee et al. (2004) focuses on the automatic detection of multiple biomedical entities using a single-word classification approach in contrast to earlier work in the area that was focusing on single entity types such as proteins or genes. In this approach, they use features such as word attributes and contextual information.

To alleviate the bottleneck of manual named entity annotation for medical texts, Yetisgen-Yildiz et al. (2010) have set up a crowdsourcing project on Amazon Mechanical Turk (MTurk)

to annotate three entity types. The research shows that using crowdsourcing is a viable alternative to annotate medical texts at scale for entity types that are understood by nonprofessionals like "medication". However, for a more complex and fine-grained distinction that requires domain knowledge, medical professionals are required. In this thesis, we specifically investigated the adaptive and interactive medical entity and relation annotation with the help of medical experts.

8.5.4 Relation Learning in the Medical Domain

EDGAR (Rindflesch et al., 2000) is a natural language processing system that extracts information about drugs and genes relevant to cancer from the biomedical literature. The entities that EDGAR focuses on are genes, cells, and drugs extracted from the MEDLINE abstracts. The system uses a statistical part-of-speech tagger for word class recognition and subsequently applies the semantic and pragmatic information to construct possible relations.

The Entity Relations (REL) task, a supporting task of the BioNLP Shared Task 2011 (Pyysalo et al., 2011), deals with the extraction of two types of **part-of relations** between a gene or protein and an associated entity. The task focused on two specific types of **object-component relations**, that hold between a gene or protein and its part (domain, regions, promoters, and amino acids) and between a protein and a complex that it is a subunit of, namely PROTEIN-COMPONENT and SUBUNIT-COMPLEX. The highest performing system achieves an F-score of 57.7%.

The work of Rosario and Hearst (2005) addresses the problem of automatically extracting protein interactions from bioscience texts. Using graphical models and a neural network, it was possible to achieve an accuracy of 64% in extracting relations from biomedical text. For training, a domain-specific database of the HIV-1 Human Protein Interaction Database containing two types of interactions, protein interactions, and human gene knock-downs (replication interactions) was employed.

While the described work constitute the state of the art of biomedical relation extraction, their level of performance is not sufficient for automatic processing. In our approach, we incorporate a human-in-the-loop approach to ensure high accuracy on a specific relation annotation that is initiated by the need of the user.

8.6 Methodology

8.6.1 Annotation Learning

The development of large amounts of high-quality training data at one shot is hard and even undesirable (Vidulin et al., 2014). Instead, an interactive and adaptive machine learning methodology is more applicable where the machine-learning model is enhanced not using the common **train-learn-evaluate** technique, but improving the model more iteratively.

An adaptive and interactive learning focuses on enhancing an embedded machine-learning model based on newly acquired training data. The benefit of adaptive learning is many-fold, such as 1) The classifier model gets better and better as new training examples are added to the training data. 2) When there is a sudden change to the underlying dataset, what is known as **concept drift**, the machine-learning model gets updated accordingly (Hoens and Chawla, 2012; Kulesza et al., 2014; Tsymbal, 2004; Žliobaitė et al., 2016). 3) It largely reduces the total annotation time required to annotate the whole dataset. Most importantly, such an approach will 4) not require a pre-existing annotation dataset so that it is truly responsive and incremental, fully adaptive to the user's need, and it makes such an approach more affordable when integrated into a more extensive information extraction system. While it is possible to use pre-existing sets of labels for entities and their relations in interactive learning, this incremental methodology 5) also allows to define and extend these label sets at any point in time during the annotation. The possibility to expand the annotation labels might be especially useful for avoiding a mismatch between the predefined labels and the need of the application.

As the machine-learning model can be enriched incrementally, applications employing this model will not be affected, as the system can still draw suggestions from the old model while building the new model. This approach overcomes the limitations where systems have to wait until full training and prediction cycles are completed, decreasing deployment time.

8.6.2 Medical NER Tagging and Relation Extraction

Medical named entity recognition is a well-researched area with a large number of datasets used in competitions (GuoDong and Jian, 2004; Kim et al., 2009, 2011; Uzuner et al., 2007, 2010). Unfortunately, biomedical annotation tasks are still challenging unlike other language process-

ing tasks since most of the annotations require highly experienced professional annotators, as we have discussed in the previous sections.

To demonstrate the effect of adaptive and interactive learning on a biomedical entity tagging, we used the BioNLP-NLPBA 2004 corpus and train a classifier using a rather generic sequence tagging system developed for German named entity recognition (Benikova et al., 2015) based on the CRFsuite (Okazaki, 2007). The system is highly configurable regarding features and data formats. In addition to the standard features (see details on the GermaNER documentation page[5]), we incorporate an automatically induced part-of-speech (POS) tag (unsupervised POS tag cluster) as a feature, which is obtained based on the unsupervised POS tagger system by Biemann (2009) that is trained on a MEDLINE 2004 dataset.

Furthermore, word shape features that reflect capitalization and character classes (e.g. numbers vs. letters), were found to be relevant for biomedical mentions, as the shape of such entities often differs from non-entity tokens. The experimental results are presented in Section 8.8.

8.7 Personalized Medical Entity and Relation Annotation Setups

8.7.1 Background for Entity Annotation

In this section, we layout setups for a personalized medical entity annotation using a human-in-the-loop (specifically the **doctor-in-loop**) approach. The setup focuses on understanding the interplay between risk factors and genetic presuppositions with leukemia cancer. The task focus on the annotation of entities in Biomedical literature, particularly for the B-chronic lymphocytic leukemia (B-CLL), a malignant hematopoietic neoplasm of B-lymphocytes (B cells), which is the most common leukemia in the westernized world (Brown, 2013)

8.7.2 Adaptive Entity Annotation

To alleviate the efforts of meaningful literature searching, we used the automation component of WebAnno designed for adaptive annotation learning. Firstly, the medical expert prepared a set of selected abstracts, downloaded from the MEDLINE. Then, based on a limited number of

[5] https://github.com/tudarmstadt-lt/GermaNER/blob/master/germaner/src/main/java/de/tu/darmstadt/lt/ner/doc/Features.md

specific medical entities, including CELL, CONDITION, DISORDER, GENE, MOLECULE, PROTEIN, MOLECULAR PATHWAY, and SUBSTANCE, the expert annotated the important structures throughout the entire text body and made them visible.

8.7.3 Entity Automation and Relation Copy Annotator

In a second setup, we took datasets from the BioNLP 2011 shared task (Kim et al., 2011) (Entity Relations Supporting Task (REL)). Our tasks include a) train a classifier for entity annotation, b) correct suggestions provided by the classifier and when appropriate add a new annotation to the dataset, and c) create a relation annotation between the existing entity annotations. In addition to the relation types specified in the BioNLP shared task, our medical expert annotated additional relation types since the existing ones were not deemed sufficient for the research question. Table 8.7.1 shows the relation types specified at the shared task and our newly added relation types.

Already at this point, we can conclude that an adaptive approach to relation extraction is more adequate to the scenario of biomedical annotation and knowledge management: Only through an adaptive approach where users can freely add new types of entities and relations, it is possible to tune the explicified information towards the user's needs. While the general-purpose setting in the BioNLP 2011 task has provided some useful relation types, it did not cover some of the relations of interest, and a static approach would require the user to re-annotate the corpus for the new relation types.

For rapid relation annotation, we have incorporated a relation copy annotator into WebAnno where relation suggestions are provided (at the lower pane in Fig. 8.7.1) as soon as annotators create relation annotations (in the upper pane in Fig. 8.7.1). This functionality has the following advantages: a) more occurrences of the same relation are automatically suggested for the remaining parts of the document and for subsequent documents, and b) an annotator can easily copy suggestions to the annotation view if the suggestions provided are correct. The impact of the relation copy annotation will be explained in the following section.

(a) Relation types from BioNLP 2011	Descriptions
Equivalent	Two Protein or Cell components are equivalent
Protein-Component	The Protein-Component is a less specific Object-Component relation that holds between a gene or protein and its component, such as a protein domain or the promoter of a gene.
Subunit-Complex	Subunit-Complex is a Component-Object relation that holds between a protein complex and its subunits, individual proteins.
(b) The new relation types	Descriptions
Activator-Reactor	Two proteins linked with the same reaction; the first one is responsible for starting the reaction and the second one responsible for its sustainability
Antibody-Antigen	An immune protein with the ability to specifically bound the antigen, a foreign substance, and to neutralise its toxicity
Cell-Marker	A set of surface proteins typical for a cell lineage or a stage of development
Cell-Variant	The main cell lineage and the subtypes which are the parts of this larger cell family
DNA-Transcript	DNA and its mRNA (messenger RNA), which translate the gene's message to a protein product
Ligand-Receptor	Two proteins or molecules, which can bind to each other because oft he complementarity of the binding site
Protein-Variant	Two proteins with the similar structure and function

Table 8.7.1: Relation types from a) the BioNLP shared task 2011 and b) identified during the relation annotation process by our medical expert.

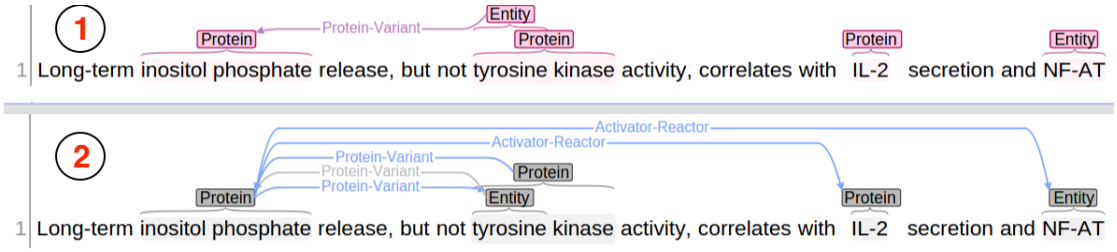


Figure 8.7.1: Relation copy annotator: Upper pane (1): relation annotation by the annotator. Lower pane (2): relation suggestions that can be copied by the user to the upper pane.

8.8 Experimental Results and Evaluations

8.8.1 Simulating Interactive Learning

To prove that adaptive and interactive machine learning can yield a quality-annotated dataset in a short training loop, we conduct our first experiment based on the BioNLP/NLPBA 2004 (Collier and Kim, 2004) dataset[6]. The dataset, which is built from 2,000 MEDLINE abstracts, is manually annotated with Biomedical technical terms. The dataset is divided into an increasing size of documents simulating interactive annotation. As it can be seen from Table 8.8.1 and Fig. 8.8.1, a (simulated) annotation of only 40 sentences already predicted an adequate amount of suggestions where users can quickly accept or modify and proceed to the next iteration. Aiming at maximizing F-score as the harmonic mean of Precision and Recall, we can clearly observe in Table 8.8.1 that, after simulated annotating of about 500 sentences, the gain in performance decreases, which implies that only annotating a small portion of the sentences produces reasonable suggestions that are mostly acceptable by the annotator. Also, we can see that more annotations beyond 5,000-10,000 sentences are subject to diminishing returns, i.e., it takes an increasing number of annotations to achieve the same amount of relative improvements, the more annotations are used for training. In a human-in-the-loop setting, this can be detected during the process and could be a sign for requiring more advanced features in the machine learning setup. This confirms our findings described in our preliminary experiment (Section 8.3), where we have reached a speedup of about a factor of 3 to 4 already with moderately accurate annotation suggestions.

8.8.2 Adaptive Entity Annotation and Relation Copy Annotator

Using the BioNLP 2011 shared task dataset, we have conducted experiments constituting two phases, i.e., entity automation and correction as well as relation annotation and suggestion.

Entity Automation: We have randomly selected 20 documents from the given training dataset (from a total of 780 documents) and train the in-built classifier using the WebAnno automation component. These documents contain 312 entity annotations, and our classifier produced 687 annotation suggestions. Later we have presented the suggestions to our medical expert to

[6] http://www.nactem.ac.uk/tsujii/GENIA/ERtask/report.html

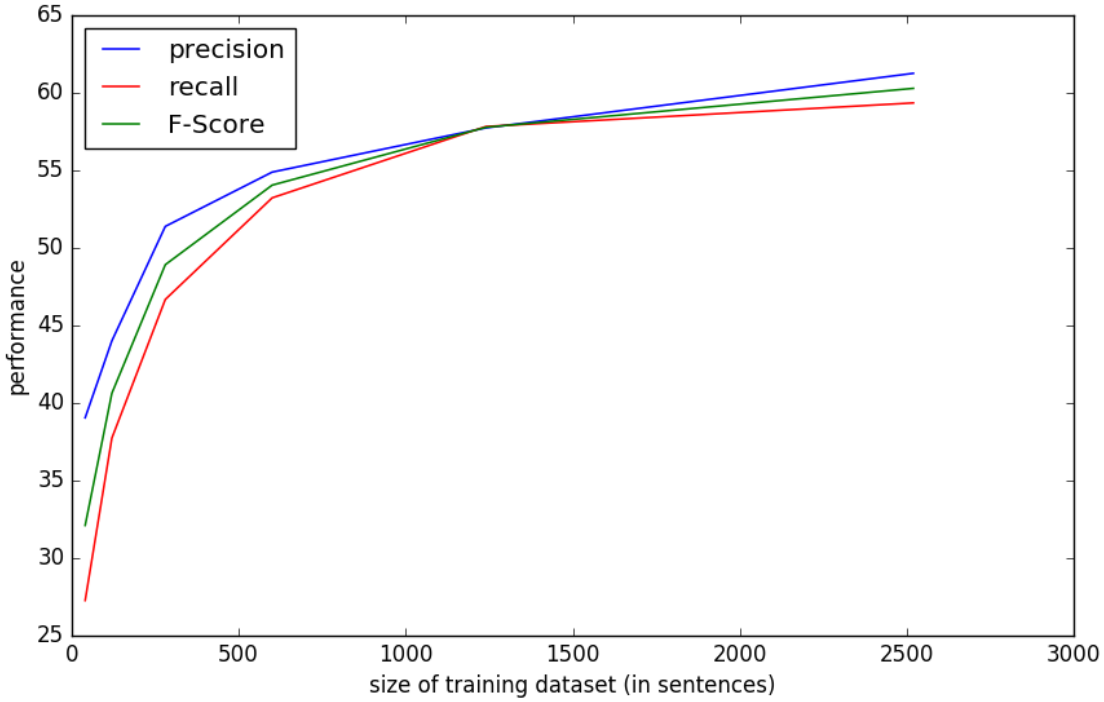


Figure 8.8.1: Learning curve showing the performance of interactive automation for BioNLP-NLPBA 2004 dataset using different sizes of training data

re-annotate the documents using the suggestion. The annotator produces a total of 752 entity annotations, in addition to the protein and entity annotations, which contains a third type of term called a **cell** that is further identified by the expert. Table 8.8.3 shows the performance of our automation system and expert annotator against the 20 documents (with gold annotations) from the BioNLP2011 REL shared task dataset.

Relation copy annotator: Once the entity annotation is completed, we have conducted relation annotation with the help of WebAnno **copy annotator**. The copy annotator produces relation suggestions in the same document where the source and target entity annotations, as well as the covered texts, match (See the details about the functionality in Chapter 3). The gold dataset contains 193 relation annotations while our annotator produces 397 relation annotations using the relation suggestions. Table 8.8.2 shows the average number of relation suggestions per document and across all documents.

We note that we can attain F-scores comparable to state of the art, which validates our ap-

Sent.	Recall	Prec.	F-score
40	27.27	39.05	32.11
120	37.74	44.01	40.63
280	46.68	51.39	48.92
600	53.23	54.89	54.05
1240	57.83	57.74	57.78
2520	59.35	61.26	60.29
5080	62.32	64.03	63.16
10200	66.43	67.50	66.96
18555	69.48	69.16	69.32

Table 8.8.1: Evaluation result for the BioNLP-NLPBA 2004 task using an interactive online learning approach with different sizes of training dataset (in number of sentences) measured in Precision, Recall and F-measure on the fixed development dataset.

Docs	All	Rels	PerRel	PerDoc	AcrossDocs
20	397	193	2.1	19.85	0.18

Table 8.8.2: Statistics of relation suggestions. For a total of 20 randomly selected BioNLP2011 REL shared task documents, there has been a total of 397 relations annotated. In the process, the system produces on average 2.1 suggestions per relation and 19.85 suggestions per document. The last column shows an average number of relation suggestions across several documents.

proach in comparison to previous approaches. More importantly, we expect a significant increase in performance when the system is used productively and can continuously extend its capabilities in long-running deployments.

8.8.3 Qualitative Assessment

In addition to the quantitative experimental simulation done in Section 8.8.1, we have conducted practical annotation and automation experiments using MEDLINE abstracts that were chosen in the context of our use case described in Section 8.7, using the automation component of WebAnno as described in Chapter 3.

The experiment was conducted in two rounds. In the first round, medical experts have annotated 5 abstracts comprising a total of 86 sentences for specific medical entities as described

Mode	Anno. Type	Recall	Prec.	F-score
Automation	Entity	61.94	49.31	54.91
	Protein	57.31	50.97	53.95
Expert	Entity	29.11	22.90	25.63
	Protein	71.94	59.28	65.00

Table 8.8.3: Machine learning automation and expert annotator performance for BioNLP 2011 REL shared task dataset.

in Section 8.7. Once the first round of annotations was completed, the automation was started using WebAnno's automation component to provide initial suggestions. As displayed in Figure 8.8.2, the automation component already suggests entity annotations immediately after the first round (some are correct and some are wrong, as it is expected).

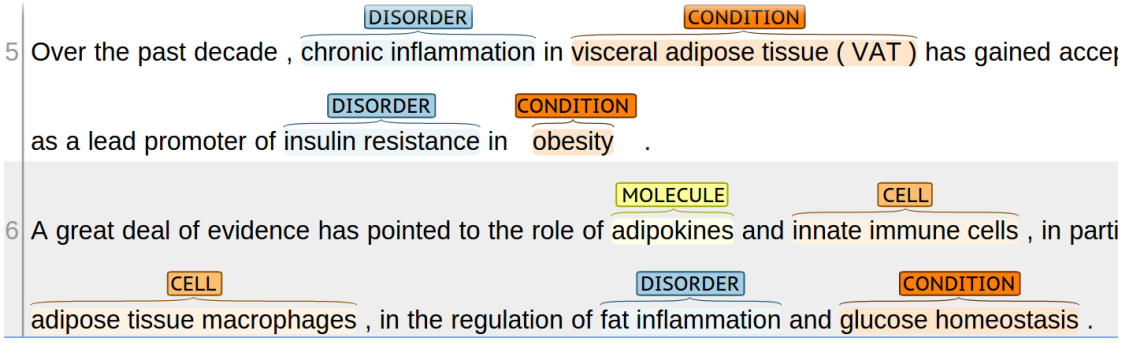
Using the automation suggestions, the expert continued annotating. After another 9 annotated abstracts that serve as training for the sequence tagging model, the quality and quantity of suggestions have again increased, see Fig. 8.8.2.

Qualitatively, annotators found that using the automation component, they perceived a significant increase in annotation speed. This confirms our results in the preliminary experiment of Amharic POS tagging automation (see Section 8.3), where adaptive annotation automation in WebAnno can speed up the annotation process by a factor of 3 to 4 in comparison to a traditional annotation interface without suggestions. On a further note, the WebAnno tool was perceived as adequate and usable by our medical professionals, requiring only minimal usage instructions.

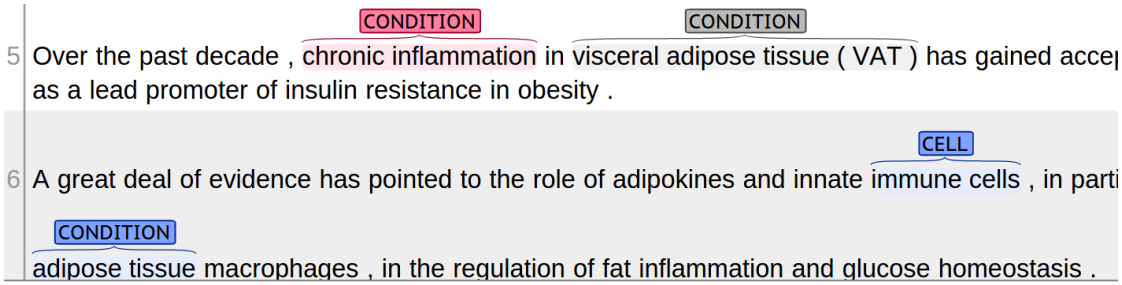
8.8.4 Analysis of the Automation and Relation Copy Annotator

As it can be seen from Table 8.8.3, on the one hand, the machine learning automation produces better performance on the general **Entity** annotation types than our expert annotator. This can be explained that the entities annotated in this dataset are very coarse level, which should be re-annotated, specifically designed to meet domain and task requirements. On the other hand, our expert annotator outperforms the automation system on the **Protein** annotation types. This is because **Protein** annotations are more specific and unambiguous to annotate.

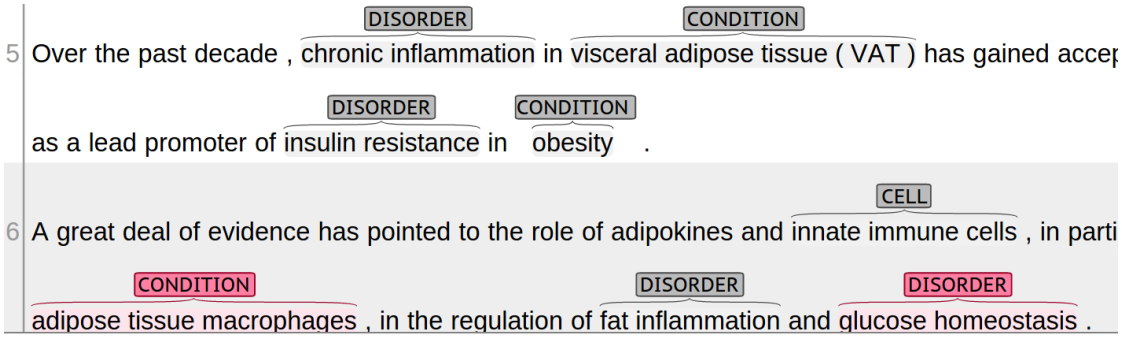
The relation copy annotator behaves as expected, as shown in Table 8.8.2, where it is possible



(a) Annotated by medical expert.



(b) Automatic suggestions after 5 abstracts are annotated.



(c) Automatic suggestions after another 9 abstracts are annotated.

Figure 8.8.2: Automation suggestions using the WebAnno automation component after annotating 5 (8.8.2b) initial resp. 9 (8.8.2c) additional abstracts. Correct suggestions are marked in grey, while wrong suggestions are marked in red. Figure 8.8.2a is the correct annotation by a medical expert.

to produce more similar relation suggestion on the same document than across several documents. We can learn from this process that 1) the low number of relation suggestion across several documents (randomly selected from the dataset) indicates that we should employ human experts in the selection of documents, which fit the domain of interest so that our system behaves as expected, and 2) a simple relation copy annotator fails to meet the need of producing adequate relation suggestions hence a proper machine learning algorithm for relation suggestion should be designed.

8.9 Conclusion

In order to validate the adaptive and interactive annotation paradigm for rapid annotation, we have first conducted preliminary experiments, where we have shown fast annotation convergence and a substantial reduction in annotation time for Amharic POS tagging and German NER.

Furthermore, we have conducted an interactive annotation experiment specifically for the biomedical domain, by employing experts in the process (**doctor-in-loop**). We investigated the impact of adaptive machine learning for the annotation of quality training data. Specifically, we tackled medical entity recognition and relation annotation on texts from MEDLINE, the largest collection of medical literature on the web.

Identifying the need of entity tagging for applications such as information extraction, document summarization, fact exploring and relation extraction, and identifying the annotation acquisition bottleneck, which is especially severe in the medical domain, we have carried out three experiments that show the utility of a **human-in-the-loop** approach for suggesting annotations to speed up the process and thus to widen this bottleneck.

In the first experimental setup, we have used an existing BioNLP-NLPBA 2004 dataset and run an experimental simulation by incrementally processing the dataset to simulate the human in the loop. Using a generic sequence tagger, we showed that annotating very few sentences already produces enough correct predictions to be useful, suggesting that adaptive and interactive annotation is a worthwhile enterprise from the beginning of an annotation project. In the second setup, we have engaged medical professionals in the annotation of medical entities in documents that were deemed relevant for the investigation of the cause of malignant B-CLL.

The automation component of the WebAnno annotation tool has been used for the annotation and automation process, and annotators found that the adaptive annotation approach 1) makes it fast and easy to annotate medical entities and 2) useful entity suggestions were already obtained after the annotation of only 5 MEDLINE abstracts, and suggestions subsequently improved tremendously after having annotated another 9 abstracts, reducing the annotation effort. The third experiment extends the same notion to relation annotation, resulting in a graph of entities and their relations per document, which gives rise to a more formalized notion of medical knowledge representation and personal knowledge management.

On a broader perspective, our results demonstrate that a paradigm change in machine learning is feasible and viable. Whereas the mantra of the past has been "there is no (annotated) data like more (annotated) data" for supervised machine learning, suggesting large annotation efforts involving many human annotators, it becomes clear from our experiments that these efforts can be sped up tremendously by switching to an approach where the human can continuously improve the model by annotation (**human-in-the-loop**) while using the model to extract information, with the especially good news that the most significant model improvements are achieved already **very early** in the process, as long as the domain is confined.

We can conclude that such an adaptive approach to machine learning that factors in the user into the equation deemed more adequate, more immediate and quickly deployable. It also fits better the shift towards an interactive, more natural, more adaptive, more contextualized, and iterative approach under the umbrella of cognitive computing.

Intelligence is the ability to adapt to change.

Stephen Hawking

9

Personalized Text Simplification using Adaptive Paraphrasing

Now, let us change our focus from a traditional way of collecting data through a dedicated annotation tool to a more likely and natural approach from the user's perspective: collecting dataset from usage data to train an embedded model in an application. As we have discussed in the previous chapters, the collection of training data is solely conducted using annotation tools, with an option of integrated automation component to make annotation process fast (the case of **rapid annotation**).

When collecting datasets using a dedicated annotation tool, one has to perform the following: 1) Study the requirements of the target application. 2) Design an appropriate annotation tool. 3) Write annotation guidelines for the annotators. 4) Deal with importing and exporting of the annotated data from or to the annotation tool. 5) Validate the quality of the collected dataset. 6)Finally develop a machine learning model and integrate it into the target application.

As we can see, this is a cumbersome process, which is costly and takes too much time. There should be a better way to minimize the processes, costs, and also deal with anticipated problems such as updating the model over time.

In this chapter, we describe an adaptive and personalized experimentation conducted for the case of adaptive text simplification using the **Par4Sem** tool that has been described in Chapter 4.

9.1 Introduction to Adaptive Text Simplification

Traditionally, training data for machine learning-based applications are collected amass, using a specific annotation tool. There are many issues regarding this approach of data collection. Most importantly, if the behavior of the target application changes over time, this makes the training data outdated and obsolete, an issue known as **concept/semantic drift**.

In this regard, we opt to design an approach where data can be collected interactively, iteratively, and continuously using an adaptive learning model in a live and real-world application. By adaptive, we mean that the learning model gets signal from the usage data through time and it automatically adapts to the need of the user based on the feedback. An adaptive learning model has a spectrum of advantages. First of all, the model gets updated continuously. The model also provides suggestions through usage, and the user can correct the suggestions, which in turn improves the model's performance. On top of this, there is no need to collect a large set of training examples a priori, which might be difficult and expensive. We also believe that instead of collecting training data in advance, it is a more natural way to get the training examples from usage data by embedding the adaptive model in the application.

In this premise, we choose an advanced NLP task, a text simplification, to experiment with an adaptive learning approach. Text simplification aims at reducing the syntactic and lexical complexity of a text for a given target reader. According to the survey by Shardlow (2014), different approaches such as lexical simplification, explanation generation, and machine translation are used for text simplification.

The main technical challenge is the integration of an adaptive paraphrasing model into a text simplification writing aid tool in order to be able to attain whether it is possible to gain NLP component quality through **adaptive learning** on **usage data**. The writing aid tool for text

simplification using an adaptive paraphrasing model consists of several components. The first component in the pipeline determines the complex or difficult words or phrases (CPs), which is based on the dataset we have discussed in Chapter 7. Once the CPs are identified, the second component produces candidate suggestions from different paraphrase resources. Candidates that do not fit the context in the sentence are filtered or excluded based on a language model score. Finally, an adaptive ranking model reorders candidates based on their simplicity and provides the ranked candidates to the user within an interactive writing aid tool.

Here, we specifically address the research questions "**How could a machine learn from usage data without explicitly obtaining training in advance?**" and "**How could a machine periodically adapt to a personalized NLP application?**" Particularly for adaptive text simplification, the main focuses are 1) How can an adaptive paraphrase ranking model be integrated into a text simplification writing aid tool?, 2) How can an adaptive paraphrase ranking model be evaluated?, and 3) Can we demonstrate the adaptivity of the approach?

In Section 9.2 we briefly review state-of-the-art works in adaptive learning and text simplification. Section 9.3 outlines how we have utilized **Par4Sem**, a semantic writing aid tool, for the experimentation of adaptive paraphrasing to the text simplification application. In Section 9.4, a brief description of the data collection and statistics of the collected data is presented. Section 9.5 describes the learning to rank machine learning algorithm used to build an adaptive ranking model, including the definition of the feature representation and the evaluation metrics employed. The experimental results obtained and the contributions of our research work are presented in Section 9.6 and Section 9.7. Finally, Section 9.8 presents the conclusions of our work and indicates future directions on the integration of adaptive approaches for NLP applications.

9.2 Related Work

The survey by Parisi et al. (2018) indicates that *continual lifelong learning*, the ability to learn continuously by acquiring new knowledge, is one of the challenges of modern computational models. The survey further explains that one of the main problems of continual learning is that training a model with new information interferes with previously learned knowledge. Žliobaitė et al. (2016) revealed that supervised machine learning approaches stationed with static datasets

face problems during deployment in a real-world application due to concept drift as applications start generating data streams continually. Applications with such properties include spam filtering and intrusion detection.

Stream-based learning and online learning (Bottou, 1998) are alternative setups to a batch-mode adaptive learning system. For example, the work by Levenberg et al. (2010) shows that the deployment of stream-based learning for statistical machine translation improves the performance of their system when new sentence pairs are incorporated from a stream. The work by Wang et al. (2015) presents SOLAR, a framework of scalable online learning algorithms for ranking, to tackle the poor scalability problem of batch and offline learning models.

Most text simplification approaches employ basic machine translation models from parallel corpora (Štajner et al., 2017; Xu et al., 2016) and using simple Wikipedia for English (Coster and Kauchak, 2011). Simple PPDB (Pavlick and Callison-Burch, 2016) is such a resource that is built automatically, using machine translation techniques on a large number of parallel corpora.

The work by Lasecki et al. (2015) shows that using crowdsourcing for text simplification is a valid approach. We also conducted our adaptive text simplification experiment on the Amazon Mechanical Turk crowdsourcing platform using a specialized text simplification tool.

The work of Bingel et al. (2018) presents **Lexi**, which is similar in setup to our adaptive approach except that 1) it explicitly deals with the text simplification task while our system can be adapted for different tasks such as academic text writing and 2) despite the detailed description of the adaptive model of the system, empirical results are not given. While Par4Sem is designed explicitly for a semantic writing aid tool, which is intended to help writers to improve their document, Lexi is implemented as a browser plugin, which is intended to help readers to understand the text better.

9.3 Par4Sem for Text Simplification

We use **Par4Sem**, an adaptive and personalized semantic writing aid tool that is described in Chapter 4 in detail, to conduct an adaptive text simplification experiment. We briefly describe the task setup here, which is relevant for text simplification. Unlike the traditional text simplification approaches, Par4Sem mimics a regular text editor (writing aid tool). The tool embeds basic text simplification functionalities such as providing suggestions for complex phrases and

editing of the text in place. Users wishing to simplify text can use the tool without building the machine learning model a priori, but the system learns from the user interactions in an iterative, adaptive, and interactive manner. In order to run a distributed and web-based simplification experiment, we have integrated the tool into a crowdsourcing platform, specifically into Amazon Mechanical Turk (MTurk).

Hence, our targeted users for the text simplification experiments are workers from the MTurk crowdsourcing platform. As a large number of workers are available in the MTurk platform, we paid special attention in the development of the tool regarding response time, accessibility, and reliability. The tool comprises a front-end component to edit texts and a back-end component, which exposes most of the requests using REST API services. Figure 9.3.1 shows the main components of Par4Sem as it is used for the text simplification task. Refer in Chapter 4 regarding how Par4Sem integrates the adaptive model. Figure 9.3.2 shows the detailed instructions presented to the crowd workers inside the MTurk browser.

While it is impossible to host complex systems inside the MTurk infrastructure, MTurk supports external human intelligence tasks (HITs) where workers can easily access our system through the MTurk interface. Par4Sem's user interface is embedded in the MTurk web page, but our server handles every activity. This design gives us full control on the collection of the dataset, for training new ranker models, and for updating the model iteratively and seamlessly while we still use the MTurk infrastructure to recruit workers and pay rewards for a web-based experiment.

As it can be seen in Figure 9.3.1, complex words or phrases (CPs) are automatically highlighted (yellow background color and underlined in cyan color). Furthermore, the users can highlight their own CPs and our system will provide ranked candidates. Besides the highlighting of CPs and providing ranked candidates, the system further provides the following functionalities specifically for the text simplification experiment.

- **Reload text**: If the worker wants to get the original text with the CPs re-highlighted, the content can be reloaded using the *Reload* button, subject to confirmation.
- **Undo and redo**: At any particular time, workers can undo or redo the changes they have made using the *Undo* and *Redo* buttons.

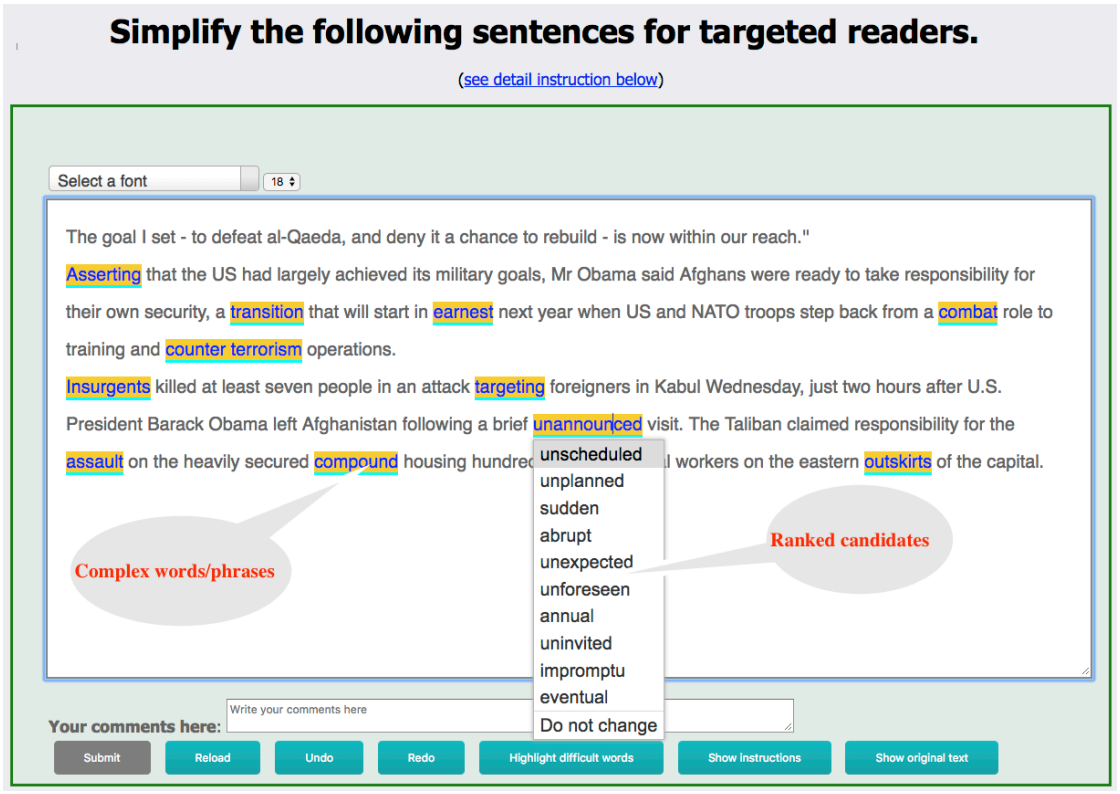
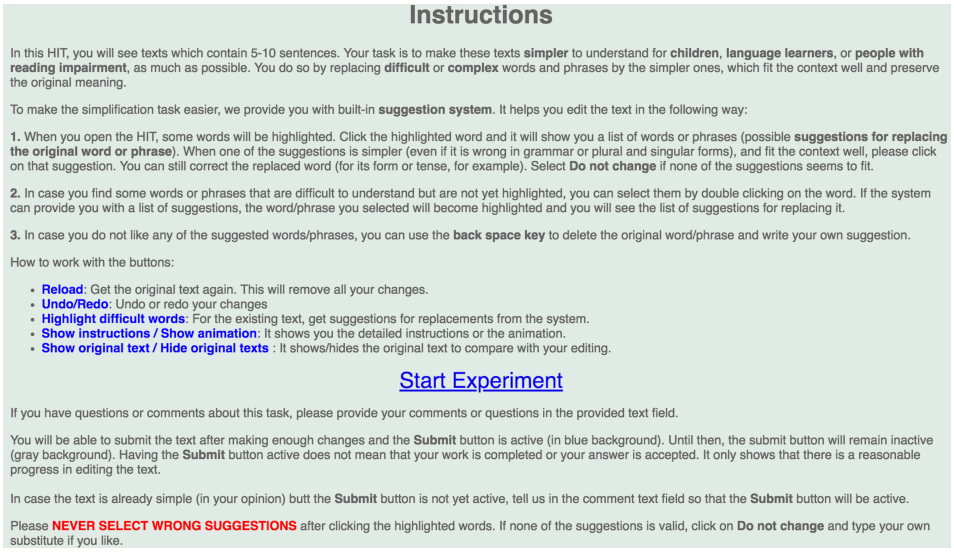


Figure 9.3.1: The Par4Sem UI as it is displayed inside the MTurk webpage with the instruction *"Simplify the following sentences for targeted readers"*. The *targeted readers* are explained in the detailed instruction as *children*, *language learner*, and *people with reading impairments*.

- **Highlight difficult words**: It is also possible to request our system to highlight difficult words automatically. This functionality is particularly essential if the worker has changed the original text, but it is not sure if the amended text is in fact simpler. Once the system highlights some words or phrases, it can still be checked if the suggestions provided by the system could still simplify the text further.

- **Show instruction / Original texts**: The instructions (see Figure 9.3.2) are visible at the beginning of each task. Once workers have accepted the task, the instructions will be hidden automatically so that the workers can focus on the simplification task in a clean window. The instruction can be displayed below the text if the worker wants to refer it



Instructions

In this HIT, you will see texts which contain 5-10 sentences. Your task is to make these texts **simpler** to understand for **children**, **language learners**, or **people with reading impairment**, as much as possible. You do so by replacing **difficult** or **complex** words and phrases by the simpler ones, which fit the context well and preserve the original meaning.

To make the simplification task easier, we provide you with built-in **suggestion system**. It helps you edit the text in the following way:

1. When you open the HIT, some words will be highlighted. Click the highlighted word and it will show you a list of words or phrases (possible **suggestions for replacing the original word or phrase**). When one of the suggestions is simpler (even if it is wrong in grammar or plural and singular forms), and fit the context well, please click on that suggestion. You can still correct the replaced word (for its form or tense, for example). Select **Do not change** if none of the suggestions seems to fit.

2. In case you find some words or phrases that are difficult to understand but are not yet highlighted, you can select them by double clicking on the word. If the system can provide you with a list of suggestions, the word/phrase you selected will become highlighted and you will see the list of suggestions for replacing it.

3. In case you do not like any of the suggested words/phrases, you can use the **back space key** to delete the original word/phrase and write your own suggestion.

How to work with the buttons:

- **Reload**: Get the original text again. This will remove all your changes.
- **Undo/Redo**: Undo or redo your changes
- **Highlight difficult words**: For the existing text, get suggestions for replacements from the system.
- **Show instructions / Show animation**: It shows you the detailed instructions or the animation.
- **Show original text / Hide original texts** : It shows/hides the original text to compare with your editing.

Start Experiment

If you have questions or comments about this task, please provide your comments or questions in the provided text field.

You will be able to submit the text after making enough changes and the **Submit** button is active (in blue background). Until then, the submit button will remain inactive (gray background). Having the **Submit** button active does not mean that your work is completed or your answer is accepted. It only shows that there is a reasonable progress in editing the text.

In case the text is already simple (in your opinion) butt the **Submit** button is not yet active, tell us in the comment text field so that the **Submit** button will be active.

Please **NEVER SELECT WRONG SUGGESTIONS** after clicking the highlighted words. If none of the suggestions is valid, click on **Do not change** and type your own substitute if you like.

Figure 9.3.2: The instructions for the text simplification task using **Par4Sem**

during the annotation task. The worker can also compare the simplified version of the text with the original text.

- **Show animation**: Crowdworkers prefer a concise task description or the task should be easy enough to be understood by most workers. To make the annotation task understandable, we decided to include an animated video showing important steps in the text simplification process. The video animation starts as soon as the task description is displayed and is hidden once the task has been started. The worker can consult the animation while completing the task.

With these functionalities, we provided a text simplification aid that comes very close to how a simplification application would look like. The goal was to provide a realistic environment in order to test the adaptive approach within a user-centric scenario. While we embedded the application into the MTurk to attract paid users, it is straightforward to provide this web-based application online or locally.

9.4 Task Description and Dataset

We specifically address text simplification, which is the task to simplify a given text that is assumed to be difficult to understand for particular readers such as language learners, children or people with reading impairments. A text simplification pipeline usually starts with the complex word or phrase identification.

To train the initial complex identification word (CWI) machine learning model, we have used parts of the dataset collected for our CWI experiment (see Chapter 7, and our paper **Yimam** et al. (2017c)), which already contains manually verified complex phrases (CP). In this dataset, the complex phrases are manually identified by 10 native and 10 non-native English speakers. The dataset has been already used for the complex word identification (CWI) shared task 2018[1](see details in our paper **Yimam** et al. (2018)).

We have generated candidate suggestions from different paraphrase resources that we discussed in Section 4.4.1.1. On top of these resources, we have also used the recently released simple PPDB (Pavlick and Callison-Burch, 2016) resource, which is a particularly relevant resource as it is built for the task of text simplification.

Apparently, the number of candidate suggestions obtained from these different resources is enormous, and we should limit the size before providing it to the ranker model. The candidates are ordered by a language model score. We trained a tri-gram language-model (Kneser and Ney, 1995) using the Wikipedia articles. The number of candidates is limited to 10; these are re-ranked using the learning to rank model.

For each HIT, we provide between 5 and 10 sentences for simplification. A HIT is then assigned to 10 different workers as we need a graded relevance to train the learning to rank model (see Section 9.5). In the experiment, we make sure that a HIT is submitted only in one iteration and most importantly, during the evaluation of the ranking model performance, we make sure that the training data from previous iterations should not contain HITs from the current iteration.

In this experiment, a total of 18,036 training instances have been collected. Figure 9.4.1 shows how the training instances appear after having collected them from the usage data. The number at the end of the simplified sentence shows the number of workers that provided the

[1] https://sites.google.com/view/cwisharedtask2018/

same simplified sentence.

Complex sentence: *Hajar said his cousin was not **affiliated** with any terrorist group.*
Simplified sentence 1: *Hajar said his cousin was not **associated** with any terrorist group.* → ***6***
Simplified sentence 2: *Hajar said his cousin was not **merged** with any terrorist group.* → ***2***
Simplified sentence 3: *Hajar said his cousin was not **aligned** with any terrorist group.* → ***1***
Simplified sentence 4: *Hajar said his cousin was not **partnered** with any terrorist group.* → ***1***

Figure 9.4.1: Examples of usage data as training instances. Here affiliated is a CP and associated, merged, aligned, and partnered are the simpler options provided by 6, 2, 1, and 1 workers respectively.

More detailed statistics are shown in Table 9.4.1. From Table 9.4.1, we see that around 70% of the workers (mainly from India and the US) have successfully completed the task.

	#complete	#visit	#total
instances	18036	10758	28794
workers	164	71	235
countries	11	3	14

Table 9.4.1: Statistics of workers and simplification instances collected during all 9 iterations in the experiment. The column **#complete** shows the number of workers who have accepted and submitted the result while the column **#visit** shows the number of workers who perform parts of the task but did not submit their results.

9.5 Learning to Rank

Learning to rank refers to a machine learning technique for training a model based on existing labels or user feedback for ranking tasks in areas like information retrieval, natural language processing, and data mining (Li, 2014). Learning to rank consists of a learning and ranking system. The system is trained by providing pairs of requests/queries and a target/ideal ranking for retrieved items. The learning model then constructs a ranking model based on the training data ranking lists.

Ranklib[2], a popular learning to rank platform in Java from the Lemur Project is used to build the ranking models. Specifically, we have used the LambdaMART algorithm to train our learning and ranking models. LambdaMART combines LambdaRank and MART (Multiple Additive Regression Trees) (Burges, 2010; Donmez et al., 2009). While MART uses gradient boosted decision trees for prediction tasks, LambdaMART uses gradient boosted decision trees using a cost function based on NDCG for solving a ranking task.

9.5.1 Machine Learning Features

To train the learning to rank model, we have designed a set of features that are important for text simplification. We have implemented the following list of features for the ranking model, which are partially derived from the candidate generating resources (Horn et al., 2014).

- **Frequency and length**: Due to the common use of these features in selecting the most simple lexical substitution candidate (Bott et al., 2012), we use three length features specifically the number of vowels, syllables, and characters and three frequency features: the frequency of the word in Simple Wikipedia, the frequency of the word in the document, and the frequency of the word in the Google Web 1T 5-Grams.

- **Lexical and distributional thesaurus resources**: We also use the number of similar words to the CPs and candidate suggestion based on lexical resources such as WordNet and a distributional thesaurus as possible features. The features are normalized and scaled using the **featran's**[3] min-max scaler tool.

- **PPDB 2.0 and simple PPDB**: From the PPDB 2.0 and simple PPDB resources, we use associated scores as given by the resource, i.e.: **ppdb2score**, **ppdb1score**, **paraphraseScore**, and **simplificationScore**.

- **Word embeddings feature**: We use the Phrase2Vec embeddings as described in Section 4.4.1.1 to obtain vector representations for targets (CPs) and candidates. The cosine similarity of the candidates with the whole sentences as well as the cosine similarity of the candidates with the tri-gram words (one word to the left and one word to the right of the

[2] https://sourceforge.net/p/lemur/wiki/RankLib/
[3] https://github.com/spotify/featran

target CP) are used as features. The vector representations of the sentences and the tri-grams are the average of the individual vector representations of words in the sentences and the tri-grams.

9.6 Learning to Rank Experiments

9.6.1 Baseline System

Our baseline system is built using a general-purpose paraphrasing dataset that we have described in Chapter 6 (see also **Yimam** et al. (2016b)). The dataset is based on essay sentences from the ANC[4] and BAWE corpora (Alsop and Nesi, 2009). We use the same feature extraction approach (see Section 9.5) for the development of the baseline model.

As it can be seen in Table 9.6.1, the results from each iteration are compared with the baseline system. We noted that the generic paraphrase datasets do not quite fit the task of text simplification as the requirements of the task are different. The lower performance of the baseline system can be attributed to the fact that the texts for the baseline system are collected from a different genre (essay sentences). We have to ensure that the first and all the subsequent iterations in the adaptive process do not use the baseline system.

9.6.2 Adaptive Systems

We start with an empty ranking model (**iteration**[5] 1), where candidates obtained from the resources are provided to the workers with a baseline ranking (based on language model scores, see Section 9.6.1). After collecting the usage data from iteration 1, we trained a ranking model, which is used to re-rank candidates for the texts in the next iteration (**iteration** 2). Texts in iteration 2, which are exclusively different from those in iteration 1 are provided to the workers. Once workers completed the simplification task at iteration 2, we re-evaluate the ranking of candidates in iteration 2 based on the usage data (using NDCG@10 metric). An NDCG@10 score of 62.88 is obtained (see Table 9.6.1), which is already better than using the baseline sys-

[4] http://www.anc.org/

[5] One iteration is where the HITs are submitted to MTurk and completed before updating the underlying model.

tem (60.66). Figure 9.6.1 shows the learning curve over the different iterations conducted in the experiment.

Similarly, training instances collected from iteration 1 and iteration 2 are used to train a ranking model, which is used to re-rank candidates for texts in the next iteration (**iteration** 3). We continued the experiment for nine iterations, and we record the performance at each iteration.

We have observed that the ranking model substantially improves in every iteration based on the NDCG@10 ranking evaluation measure. Table 9.6.1 also shows that if we test the models from earlier iterations, the performance of the system declines. Thus, the system can make good use of more usage data if available. For example, on iteration 6, testing on a ranking model that is trained based on training instances from iteration 1 up to iteration 5 ($\leq$5) produces an NDCG@10 score of 72.36 while testing on the ranking model trained based on training instances from iteration 1 up to iteration 4 ($\leq$4) produces an NDCG@10 score of 69.88.

Our result confirms all our initial hypotheses and addresses our research questions we have stated in Chapter 1, particularly **RP3** (”**How can NLP applications get training data from application usage without employing annotation tools?**” and ”**How to build personalized machine learning models for an NLP application using the human-in-the-loop paradigm?**”). First, integrating an adaptive ranking model works very well as can be seen from Table 9.6.1 and Figure 9.6.1. Secondly, we can witness that a model trained on the datasets obtained using the adaptive paraphrasing model performs way better than using generic paraphrase ranking models (the baseline system).

Furthermore, we have explored the effect of the adaptive system for individual workers (**user adaption**). In this case, we have built simulated unique models for the 10 top workers, who have participated in at least 4 different batches (iterations) of the task. We use the first iteration dataset where only the worker has participated in (we do not use usage data from other workers) to train an initial model, and start using the model for the subsequent iterations (building a **personalized model**). As we can see from Table 9.6.2 and Figure 9.6.2, the NDCG scores improve consistently over the iterations. The results also revealed that text simplification could be modeled differently based on the need of the user (**personalization**).

Testing	NDCG@10									
	Training instances on previous iterations									
	#sentences	baseline	1	≤ 2	≤ 3	≤ 4	≤ 5	≤ 6	≤ 7	≤ 8
1	115	-	-	-	-	-	-	-	-	-
2	214	60.66	62.88	-	-	-	-	-	-	-
3	207	61.05	63.39	65.52	-	-	-	-	-	-
4	210	58.21	60.73	65.93	67.46	-	-	-	-	-
5	233	56.10	62.53	65.66	66.00	70.72	-	-	-	-
6	215	62.18	61.05	66.51	67.86	69.88	72.36	-	-	-
7	213	57.00	62.07	64.02	64.88	67.28	69.27	74.14	-	-
8	195	56.56	59.53	62.11	63.03	64.54	67.40	71.05	75.83	-
9	224	56.14	63.48	65.58	65.87	69.18	69.51	71.31	71.40	75.70

Table 9.6.1: NDCG@10 results for each iteration of the testing instances using training instances from the previous iteration. For example, for testing at iteration 2, the NDCG@10 result using training data from the previous iteration, i.e. iteration 1, is 62.88. The baseline column shows the performance in each iteration using the generic paraphrasing dataset used to train the baseline ranking model.

	Iteration 2			Iteration 3			Iteration 4		
Workers	Instances (#)	positive (%)	NDCG score	Instances (#)	positive (%)	NDCG score	Instances (#)	positive (%)	NDCG score
AXXXL5	950	10.21	51.35	2661	10.11	55.87	2771	9.78	56.57
AXXX3N	1591	10.31	45.45	3130	10.29	48.72	5367	10.23	47.98
AXXXMY	1117	10.12	55.15	2753	10.10	61.35	4809	10.13	63.76
AXXXI7	70	10.00	49.33	2162	10.59	64.10	3988	10.38	66.82
AXXX56	1190	10.42	54.63	2468	10.29	56.24	4477	10.27	58.79
AXXXS1	824	10.19	54.45	1845	10.24	55.28	3045	10.15	58.78
AXXXM9	448	10.04	55.25	896	10.04	56.00	2669	11.09	58.61
AXXXAM	1594	10.16	60.59	2999	10.17	61.96	4611	10.13	63.28
AXXX3E	615	10.73	59.44	1038	10.69	59.51	3451	10.66	62.59
AXXXGI	100	24.00	45.05	1979	11.22	56.72	3160	10.79	57.35

Table 9.6.2: The NDCG result for 10 different workers. *Instances* shows the total number of training instances used from the previous iteration (only for the respective user) while *positive* shows the total number of positive feedback provided by the user. The workers' ID is obscured to protect their privacy.

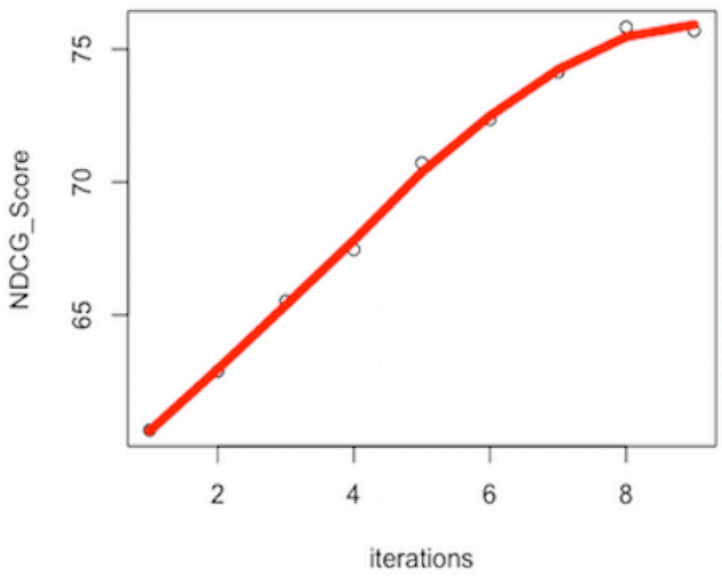


Figure 9.6.1: Learning curve showing the increase of NDCG@10 score over 9 iterations.

9.7 Discussion

Most text simplification systems, and for that matter, most NLP models, are based on a traditional **collect and train** approach where all the required training data are annotated first, and then training and evaluation is carried out after the data collection. This experiment is the first scientific work (**Yimam** and Biemann, 2018) to conduct an adaptive approach for text simplification where signals from usage data are collected in an adaptive, interactive, and iterative approach improving the model of an NLP component.

We have demonstrated that the adaptive approach has many contributions: 1) the adaptive learning model is integrated into a practical NLP application (**live-usage**), 2) the performance of the integrated adaptive model improves through usage data of the NLP application (**adaptability**), 3) the integrated learning model potentially adapts to the needs of the user or user groups through usage data (**personalized NLP**), and 4) we also have shown that adaptive

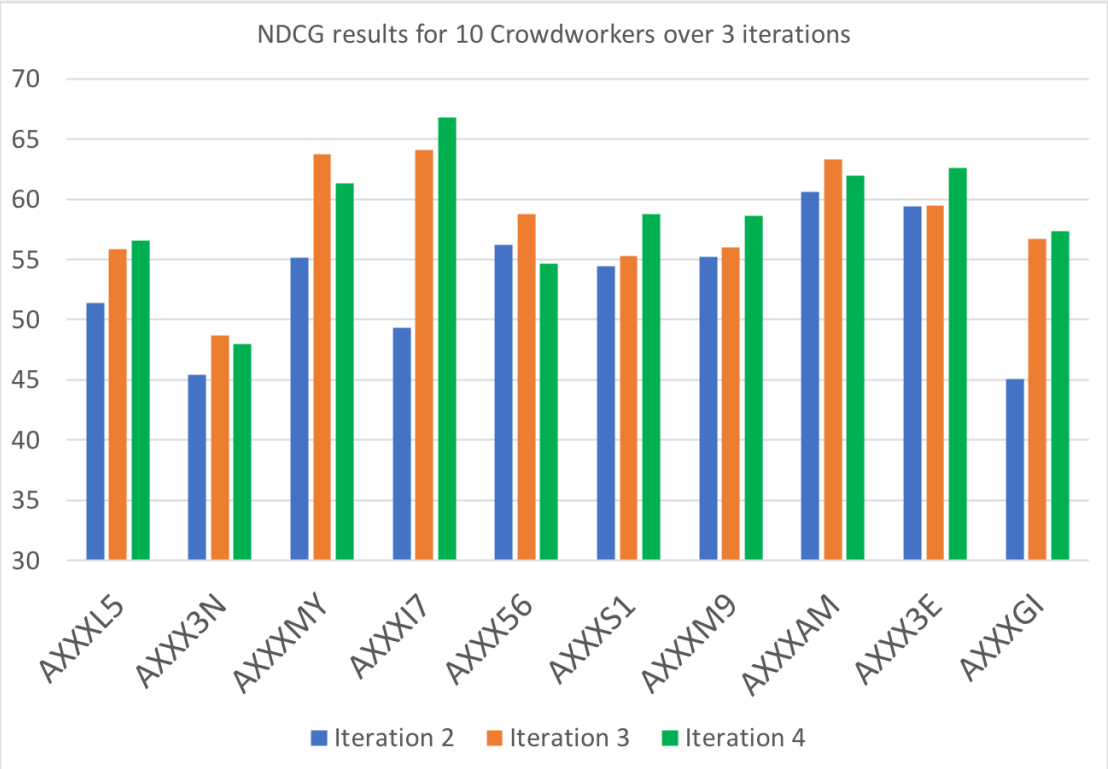


Figure 9.6.2: The increase of NDCG scores over 3 iterations for the top 10 workers ordered by their productiveness (who have completed most HITs over several iterations).

systems can be evaluated incrementally, by comparing the system's suggestions by the ranking model to the actual ranking provided by the users (**incremental evaluation**).

In this research, we also have showcased how to perform web-based and real-time adaptive data collection using the Amazon MTurk crowdsourcing platform. The MTurk crowdsourcing platform has been mainly used to collect datasets for tasks that are not complex and difficult to complete such as identifying named entities or biomedical entities in text, categorizing texts for spam, labeling an image with appropriate captions and so on. Using MTurk's external HIT, we have shown that the MTurk crowdsourcing platform can be successfully used for complex NLP applications such as text simplification with a writing aid tool, which usually is limited to a lab-based experiment.

9.8 Conclusion

In this chapter, we have shown that the integration of an adaptive paraphrase ranking model effectively improves the performance of text simplification task. We have designed a full-fledged, web-based based text simplification system called **Par4Sem** where we have integrated an adaptive paraphrase ranking model into the tool.

Our tool is integrated with the Amazon Mechanical Turk crowdsourcing platform to collect usage data for text simplification.

To evaluate the performance of the adaptive system on the collected usage data, we have evaluated the ranking model performance in an iterative way. In every iteration, we use the usage data exclusively from the previous iterations (except the first iteration that is used solely as training data and we do not evaluate it) to train the learning to rank model. The result shows that, in every iteration, there is a substantial increase in performance based on the NDCG@10 evaluation metric.

This experiment demonstartes how to develop a personalized NLP application. Using a similar approach, one can effectively deploy **Par4Sem** for a different purpose such as to write technical documents. The research also sheds light on domain or task adaptions. One can use datasets collected for general purpose domains, and it is possible to adapt the model based on the usage data over a period of time. This is a much cheaper alternative than collecting a special-purpose labeled dataset from scratch.

In the future, we would like to run a long-term study with arbitrary users and arbitrary texts using **Par4Sem**, which is a freely available online tool for text composing. Besides, we would like to investigate if our approach can be extended to syntactic simplification. We also envision further possible tasks where adaptive learning can be helpful, such as collaborative text composing and recommender systems.

Part V

Conclusion and Future Directions

... Times and conditions change so rapidly that we must keep our aim constantly focused on the future.

Walt Disney

10

Conclusion and Future Directions

10.1 Conclusion

In this thesis, two main achievements have been accomplished: 1) the rapid annotation paradigm for the collection of datasets for natural language processing (NLP) applications, and 2) the integration of adaptive and personalized models into NLP applications. Most NLP applications, such as question answering, machine translation, text simplification, text summarization, and sentiment analysis require a machine learning component that relies on training data. The collection of training data is generally expensive and takes a considerably large amount of time to complete the task.

The rapid annotation paradigm we have attained is meant to achieve the following: 1) Substantially reduce the annotation time, 2) reduce the cost of the annotation, and 3) assist the annotator in understanding the annotation by developing a simple and intuitive annotation tool.

While the rapid annotation paradigm already helps a lot in collecting quality datasets to build

a better machine learning model for NLP applications, we further investigated how to integrate an adaptive and personalized model into NLP applications. The primary objective of this approach is to start the NLP application with existing training datasets or optionally without training examples and to allow continuous collection of training examples from usage data to adapt and update the model. In essence, the NLP application integrates a machine learning model that barely helps at the beginning but provides life-long learning and model updating capabilities.

10.2 Detailed Contributions

Background of the Study

In Chapter 2, we have briefly presented the characteristics of different annotation processes, how traditional annotation tools are developed, and the main problems associated with the annotation tools. We also presented different types of annotation tools, and the different types of annotation formats suggested in related work. A brief introduction into the rapid annotation paradigm and the integration of adaptive models into NLP applications has also been given. We can conclude from this that there is barely any previous research on adaptive NLP applications.

Annotation, Visualization, and Adaptive NLP Tools

In Chapter 3, Chapter 4, and Chapter 5, we presented different tools that have been developed either to support rapid annotation or to conduct adaptive and personalized NLP application experiments.

In Chapter 3, we presented **WebAnno**: a web-based, distributed, and generic annotation tool that we have partially developed in the scope of this thesis. While it has established itself as one of the most popular generic annotation tools in the NLP community mostly for its standard annotation functionality, it also exhibits most of the rapid annotation paradigm we have envisioned. The most notable rapid annotation features implemented in WebAnno include:

1. Designing and implementing an easy-to-use user interface that mainly includes creating annotations, importing and exporting annotated documents, monitoring the annotation progress, and adjudicating annotations.

2. Reducing hand movement during annotation that includes: scrolling pages and documents only with keyboard shortcuts, automatically forwarding annotation positions, choosing annotation labels with suggestive key-strokes, automatically adding labels or tags into an existing tag-set.

3. Suggestion and correction capabilities that allow to import externally annotated or labelled data and allow annotators to make corrections easily, repeating the same annotation in context to the whole document in a project, building a machine learning model based on training examples (either imported from external sources, using the annotations already available in WebAnno, or both).

In Chapter 4, we have presented the design approaches for the development of an adaptive NLP application, mainly a semantic writing aid tool. Writing aids seem a natural candidate for an adaptive NLP application because existing writing systems already depend on suggestions from static resources such as thesauri, but these resources mostly do not consider context and do not extend the underlying resources based on usage data.

Our semantic writing aid tool called **Par4Sem** embeds an adaptive and personalized machine learning model that relies on usage data to train the model incrementally and continuously. The adaptive model is specifically used to present paraphrases and learns how to rank the paraphrases based on the user interaction. For example, if the user is employing the writing aid to simplify documents, that is, producing a similar text that is easier to understand for a given target reader, such as children, language learners, or people with reading impairments, the tool learns how to re-write such text based on the history of interactions (usage data) collected.

In Chapter 5, we have presented how to use visualization tools to collect training examples. Unlike annotation tools, visualization tools are usually meant to present information using visual components to easily understand the underlying data and draw a fact out of it. We presented how to produce training examples based on usage from the visualization tools. **New/s/leak** was a tool developed for investigative data journalists to find out interesting stories out of huge datasets, mainly leaked texts such as Wikileaks[1], the Panama Papers[2], and the Enron mail dataset[3].

[1] https://wikileaks.org/
[2] https://www.icij.org/investigations/panama-papers/
[3] https://www.cs.cmu.edu/~./enron/

While journalists use the tool for investigative and explorative operations, it further allows collaborative document editing, labeling and correcting entities, which in the end might be used to enhance the underlying machine learning model.

Similarly, the **Network of the day** (NoD) is a tool to visualize interesting and trending facts of a day, centered around important participating entities extracted from daily released newswire articles. The network graph, which is clustered based on the relations of participating entities, shows what has happened on that day by presenting a network of associated entities. During the network navigation and exploration, it is possible to add comments and labels about a given relation between two entities. The collected data can be used to train a machine learning model that can automatically infer possible relations among entities in text.

We conclude that for user-facing adapting tools, an interface that naturally supports the interactions needed for data collection is crucial.

Resources for Text Paraphrasing

In the process of developing an adaptive and personalized NLP application, we were particularly interested in confirming the validity of the approach. In this regard, we have chosen text simplification, a particular NLP application that deals with providing a simpler version of a text for particular readers. For this process, we have specifically collected resources for complex word identification and paraphrase ranking. A complex word or phrase identification is a subtask where texts are first analyzed, and difficult words are identified. In Chapter 7, we have discussed how we collect and experiment with multilingual and cross-lingual complex word identification datasets. In Chapter 6, we have presented the different approaches we follow to collect paraphrase resources that have been used as a baseline system for general paraphrasing ranking systems.

From thee works, we can conclude that a) paraphrasing is indeed depending on context and b) crowdsourcing is a viable way of collecting respective datasets.

Rapid Annotation Experiments

In Chapter 8, we have presented three use-cases for adaptive annotation experiments with

WebAnno. For each of them, we have evaluated the impact of adaptive annotation regarding annotation speed and dataset quality. In the first use-case, we have conducted annotation automation for Amharic part-of-speech tagging and have evaluated the total reduction of time to complete the annotation. We have found that the automation component shortens the total annotation time approximately by a factor of 3 to 4 compared to the traditional annotation process.

In the second experimental setup, we have demonstrated the performance improvement over time by simulating German named entity recognition in a continuous manner. We found that, at around 3% of the training data (using only 696 out of $24{,}000$ sentences), an F-score of 53.40% can be achieved compared to an F-score of 74.05% when the whole dataset is used. The simulation confirms the benefit of using an adaptive and iterative annotation approach where a more significant gain is already attained at the earlier stage of the annotation process.

In the third experiment, we use the automation component to annotate real-world and domain-specific annotation problems. The experiment is conducted for a Biomedical domain with experts in the field (**doctor-in-loop**). In addition to evaluating the performance of the incremental models, we also collect expert's feedback towards the automation component. The medical expert was able to identify her own annotation problem, by collecting medical abstracts and annotating specific medical entities. In this regard, the automatic annotation component helps the expert 1) in providing enough suggestions of medical entities after initial use of the system with very few abstracts, 2) in suggesting relation annotation between medical entities based on previous relation annotations.

We conclude that the rapid annotation setup should be preferred, as it gives consistently quicker turnarounds, in all situations where the annotation can indeed be automated. This at least requires concise annotation guidelines and a strong correlation of annotations with textual elements.

Adaptive and Personalized NLP applications

In Chapter 9, we have discussed the practical experiments with an adaptive NLP application. Using the **Par4Sem** semantic writing aid tool, a series of experiments are conducted to investigate if the adaptive model helps in re-ranking paraphrases for text simplification. Par4Sem was

integrated into the Amazon Mechanical Turk (MTurk) crowdsourcing platform, where workers are requested to simplify a given text. The experiment has been conducted in multiple iterations, where the feedback obtained from the previous iterations was used to update the adaptive paraphrasing model. The experiment has proved that 1) the paraphrase ranking model is improved continuously based on the usage data and 2) the model can also adapt to individual text simplification needs.

This marks the main contribution of this thesis, as to the best of our knowledge, this is the first time that a semantic NLP component is improved through mere usage and the dataset connection is done in an implicit, not explicit fashion.

10.3 Limitations and Open Issues

Concept/semantic drift: In this thesis, we tackle the problem of **concept/semantic drift** using a **life-long-learning** approach. Since the model is updated continuously with new training examples, it will properly deal with new concepts arising over time. However, we do not explore yet how to **unlearn** old concepts, or do we need at all to forget such concepts? How do we know the time in point to forget or unlearn a given concept? This is an open issue to deal with.

Relation adaption: We use a straightforward ad-hoc approach for the adaptive relation annotation. While our approach confirms that relation adaption works well when the documents to be annotated are similar enough, it fails short to produce enough suggestions when the documents are not related. However, we have not investigated whether a machine learning based relation automation solves this problem or not.

Complexity vs. comprehensibility: The complex word identification task mainly focuses on determining the complexity level of lexical words or phrases. The text simplification task we have carried out is based on the model built from the CWI datasets. In the scope of this thesis, we do not investigate, if the simplified text is comprehensible or not. This is an open issue that we do not explore in this thesis, but we indicated how to deal with it as possible future work in Section 10.4.

10.4 Future Directions

The rapid annotation experimental results for biomedical annotation are promising. In addition to the annotation task, we can see that integrating the adaptive model for real-world biomedical systems is a possibility.

The text simplification dataset we have collected can be further used to study text comprehensibility. We do not know yet whether the simpler texts that are produced based on the complex word identification and the paraphrase ranking components are more comprehensible or not. One possible approach is, to present both complex text (original) and the simplified text to different users, for example, different crowd workers, and request them to determine which of the two texts is more understandable.

The adaptive NLP application experiment we have conducted can also be extended for different languages other than English. To experiment with different languages, the only required dataset are paraphrase resources. PPDB provides paraphrase resources for Arabic, Chinese, German, French, Italian, Polish, and Spanish in addition to English, which makes running experiments for these languages easier.

The design of our writing aid tool uncovers much more comprehensive research directions. One possible direction is to learn how to adapt to a specific writing style over time. Existing writing aids, for example, the thesaurus-based paraphrase suggestions support in most word processors or the syntactic and semantic support in online tools such as Grammarly[4] tool assume a somewhat consistent style of writing, which falls short of adapting to the user's needs.

The adaptive tool can also be used by language learners, where it helps the learner to expand their vocabulary. Unlike the text simplification task, the adaptive system should learn how to provide new vocabulary (hence difficult words) from the suggestions. This is effectively reversing the CWI system, which learns how to replace a simple word or phrase with a more difficult candidate.

Furthermore, a full-fledged adaptive text simplification system, which incorporates a syntactic simplification task followed by lexical simplification, should also be explored. Before applying lexical simplification, the system could learn whether the sentence needs syntactic simplification or not. It might be cumbersome to make the whole syntactic simplification task (the

[4] https://grammarly.com

analysis, transformation, and regeneration components) (Siddharthan, 2006) adaptable, but at least some parts of the task, such as determining when and where should a sentence be split could be learned from aggregating user actions on the syntactic level.

References

Ahmed AbuRa'ed and Horacio Saggion. LaSTUS/TALN at Complex Word Identification (CWI) 2018 Shared Task. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, LA, USA, 2018. Online: http://aclweb.org/anthology/W18-0517.

David Alfter and Ildikó Pilán. SB@GU at the Complex Word Identification 2018 Shared Task. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, LA, USA, 2018. Online: http://aclweb.org/anthology/W18-0537.

Sian Alsop and Hilary Nesi. Issues in the development of the British Academic Written English (BAWE) corpus. *Corpora*, 4(1):71–83, 2009. Online: https://www.researchgate.net/publication/250229334_Issues_in_the_development_of_the_British_Academic_Written_English_BAWE_corpus.

Sandra M. Aluísio, Lucia Specia, Thiago A.S. Pardo, Erick G. Maziero, and Renata P.M. Fortes. Towards Brazilian Portuguese automatic text simplification systems. In *Proceedings of the eighth ACM symposium on Document engineering*, pages 240–248, Sao Paulo, Brazil, 2008. Online: http://rgcl.wlv.ac.uk/papers/Specia_DocEng2008.pdf.

Marcelo Adriano Amancio and Lucia Specia. An Analysis of Crowdsourced Text Simplifications. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 123–130, Gothenburg, Sweden, 2014. Online: http://aclweb.org/anthology/W14-1214.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. Massively multilingual word embeddings. *CoRR*, abs/1602.01925, 2016. Online: https://arxiv.org/pdf/1602.01925.pdf.

Mykhaylo Andriluka, Jasper R. R. Uijlings, and Vittorio Ferrari. Fluid Annotation: A Human-Machine Collaboration Interface for Full Image Annotation. In *Proceedings of the 26th ACM*

International Conference on Multimedia, pages 1957–1966, Seoul, Republic of Korea, 2018. ISBN 978-1-4503-5665-7. Online: http://doi.acm.org/10.1145/3240508.3241916.

Ion Androutsopoulos and Prodromos Malakasiotis. A Survey of Paraphrasing and Textual Entailment Methods. *Journal of Artificial Intelligence Research*, 38(1):135–187, May 2010. ISSN 1076-9757. Online: https://arxiv.org/abs/0912.3747.

Segun Taofeek Aroyehun, Jason Angel, Daniel Alejandro Pérez Alvarez, and Alexander Gelbukh. Complex Word Identification: Convolutional Neural Network vs. Feature Engineering . In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, LA, USA, 2018. Online: http://aclweb.org/anthology/W18-0538.

Ramy K. Aziz, Daniela Bartels, Aaron A. Best, Matthew DeJongh, Terrence Disz, Robert A. Edwards, Kevin Formsma, Svetlana Gerdes, Elizabeth M. Glass, Michael Kubal, Folker Meyer, Gary J. Olsen, Robert Olson, Andrei L. Osterman, Ross A. Overbeek, Leslie K. McNeil, Daniel Paarmann, Tobias Paczian, Bruce Parrello, Gordon D. Pusch, Claudia Reich, Rick Stevens, Olga Vassieva, Veronika Vonstein, Andreas Wilke, and Olga Zagnitko. The RAST Server: Rapid Annotations using Subsystems Technology. *BMC Genomics*, 9(1):75, 2008. ISSN 1471-2164. Online: https://bmcgenomics.biomedcentral.com/articles/10.1186/1471-2164-9-75.

Ricardo Baeza-Yates, Luz Rello, and Julia Dembowski. Cassaurus: A resource of simpler spanish synonyms. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 951–955, Portorož, Slovenia, 2016. Online: https://www.superarladislexia.org/pdf/2016-Baeza-YatesRello-Cassarus-lrec.pdf.

Timothy Baldwin and Aline Villavicencio. Extracting the unextractable: A case study on verb-particles. In *Proceedings of the 6th Conference on Natural Language Learning*, pages 1–7, Taipei, Taiwan, 2002. Online: http://aclweb.org/anthology/W02-2001.

Kathrin Ballweg, Florian Zouhar, Patrick Wilhelmi-Dworski, Tatiana von Landesberger, Uli Fahrer, Alexander Panchenko, **Seid Muhie Yimam**, Chris Biemann, Michaela Regneri, and Heiner Ulrich. New/s/leak – A Tool for Visual Exploration of Large Text Document Collections in the Journalistic Domain. In *VIS conference 2016 collocated with the Workshop on Visualisation in Practice*, Baltimore, MD, USA, 2016. Online: http://www.aclweb.org/anthology/P16-4028.

Colin Bannard and Chris Callison-Burch. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604, Ann Arbor, MI, USA, 2005. Online: http://aclweb.org/anthology/P05-1074.

Daniel Beck, Lucia Specia, and Trevor Cohn. Reducing annotation effort for quality estimation via active learning. In *Proceedings of ACL 2013 System Demonstrations*, Sofia, Bulgaria, 2013. Online: http://www.aclweb.org/anthology/P13-2097.

Darina Benikova, Chris Biemann, and Marc Reznicek. NoSta-D Named Entity Annotation for German: Guidelines and Dataset. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, Reykjavik, Iceland, 2014a. Online: https://pdfs.semanticscholar.org/87dc/ed7d3aa2a3ebe270bfeca13db5708d9537ce.pdf.

Darina Benikova, Uli Fahrer, Alexander Gabriel, Manuel Kaufmann, **Seid Muhie Yimam**, Tatiana von Landesberger, and Chris Biemann. Network of the Day: Aggregating and Visualizing Entity Networks from Online Sources. In *Proceedings of NLP4CMC at KONVENS2014*, pages 48–52, Hildesheim, Germany, 2014b. Online: https://hildok.bsz-bw.de/files/277/01_07.pdf.

Darina Benikova, **Seid Muhie Yimam**, Prabhakaran Santhanam, and Chris Biemann. GermaNER: Free Open German Named Entity Recognition Tool. In *Proceedings of GSCL 2015*, pages 31–28, Essen, Germany, 2015. Online: https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2015-benikovaetal-gscl2015-germa.pdf.

Chris Biemann. Semantic indexing with typed terms using rapid annotation. In *the TKE-05-Workshop on Methods and Applications of Semantic Indexing*, Copenhagen, Denmark, 2005. ISBN 0251-5253. Online: https://pdfs.semanticscholar.org/8d98/f7d7c3278b3ee4cc9f67f74e85ece386c7d3.pdf.

Chris Biemann. Unsupervised Part-of-Speech Tagging in the Large. *Research on Language and Computation*, 7(2):101–135, 2009. ISSN 1572-8706. Online: https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2009-biemann-rolc-unsupose.pdf.

Chris Biemann. Creating a System for Lexical Substitutions from Scratch using Crowdsourcing. *Language Resources and Evaluation: Special Issue on Collaboratively Constructed Language Resources*, 46(2), 2012. eBook link: http://www.springerlink.com/content/07v735p2p1202xn2/.

Chris Biemann. Design principles for transparent software in computational humanities. In *Report from Dagstuhl Seminar 14301: Computational Humanities - bridging the gap between Computer Science and Digital Humanities*, pages 88–91. Dagstuhl Publishing, Germany, 2014. Online: https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2014-biemann-dagstuhl.pdf.

Chris Biemann and Martin Riedl. Text: Now in 2D! A Framework for Lexical Expansion with Contextual Similarity. *Journal of Language Modelling*, 1(1):55–95, 2013. Online: https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2013-biemann-riedl-jlm-text2d.pdf.

Chris Biemann, Kalina Bontcheva, Richard Eckart de Castilho, Iryna Gurevych, and **Seid Muhie Yimam**. Collaborative Web-Based Tools for Multi-layer Text Annotation. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, pages 229–256. Dordrecht, 2017. preprint: https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2017-biemannetal-hola-webbasedtools-preprint.pdf.

Joachim Bingel and Johannes Bjerva. Cross-lingual complex word identification with multitask learning. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, LA, USA, 2018. Online: http://www.aclweb.org/anthology/W18-0518.

Joachim Bingel, Gustavo Paetzold, and Anders Søgaard. Lexi: A tool for adaptive, personalized text simplification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 245–258, Santa Fe, NM, USA, 2018. Online: http://aclweb.org/anthology/C18-1021.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research (JMLR)*, 3:993–1022, 2003. Online: http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf.

Kalina Bontcheva, Hamish Cunningham, Ian Roberts, and Valentin Tablan. Web-based collaborative corpus annotation: Requirements and a framework implementation. In *New Challenges for NLP Frameworks workshop at LREC-2010*, Valletta, Malta, 2010. Online: https://gate.ac.uk/sale/lrec2010/teamware/teamware-lrec10.pdf.

Kalina Bontcheva, Hamisch Cunningham, Ian Roberts, Angus Roberts, Valentin Tablan, Niraj Aswani, and Genevieve Gorrell Gorrell. GATE Teamware: a web-based, collaborative text annotation framework. *Language Resources and Evaluation*, 47(4):1007–1029, 2013. Online: https://link.springer.com/article/10.1007/s10579-013-9215-6.

Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3 Data-Driven Documents. *IEEE Trans. Visualization & Comp. Graphics (Proceedings of InfoVis)*, 17(12):2301–2309, 2011. Online: http://vis.stanford.edu/files/2011-D3-InfoVis.pdf.

Stefan Bott, Luz Rello, Biljana Drndarevic, and Horacio Saggion. Can Spanish be simpler? LexSiS: Lexical simplification for Spanish. In *Proceedings of COLING 2012*, pages 357–374, Mumbai, India, 2012. Online: http://aclweb.org/anthology/C12-1023.

Léon Bottou. On-line Learning in Neural Networks. chapter On-line Learning and Stochastic Approximations, pages 9–42. New York City, NY, USA, 1998. ISBN 0-521-65263-4. Online: http://bit.do/onlineBottou.

Jordan Boyd-Graber and David M. Blei. Multilingual Topic Models for Unaligned Text. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 75–82, Montreal, QC, Canada, 2009. Online: https://arxiv.org/pdf/1205.2657.pdf.

Chris Brockett and William B. Dolan. Support Vector Machines for Paraphrase Identification and Corpus Construction. In *Third International Workshop on Paraphrasing (IWP2005)*, pages 1–8, Jeju Island, South Korea, 2005. Online: http://aclweb.org/anthology/I05-5001.

Laetitia Brouwers, Delphine Bernhard, Anne-Laure Ligozat, and Thomas François. Syntactic Sentence Simplification for French. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 47–56, Gothenburg, Sweden, 2014. Online: https://www.aclweb.org/anthology/W14-1206.

Jennifer R. Brown. Inherited susceptibility to chronic lymphocytic leukemia: evidence and prospects for the future. *Therapeutic advances in hematology*, 4(4):298–308, 2013. Online: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3734903/.

Christopher J. C. Burges. From RankNet to LambdaRank to LambdaMART: An Overview. Technical report, Microsoft Research, 2010. Online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.180.634&rep=rep1&type=pdf.

Steven Burrows, Martin Potthast, and Benno Stein. Paraphrase Acquisition via Crowdsourcing and Machine Learning. *ACM Transactions on Intelligent Systems and Technology*, pages 43:1–43:21, 2013. Online: https://webis.de/downloads/publications/papers/stein_2013c.pdf.

Andrei Butnaru and Radu Tudor Ionescu. UnibucKernel: A kernel-based learning method for complex word identification. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, LA, USA, 2018. Online: http://aclweb.org/anthology/W18-0519.

John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. Simplifying text for language-impaired readers. In *Proceedings of EACL 1999*, pages 269–270, Bergen, Norway, 1999. Online: http://www.aclweb.org/anthology/E99-1042.

Wei-Te Chen and Will Styler. Anafora: A Web-based General Purpose Annotation Tool. In *Proceedings of the NAACL HLT 2013 Demonstration Session*, pages 14–19, Atlanta, GA, USA, 2013. Online: http://www.aclweb.org/anthology/N13-3004.

Zhilu Chen and Xinming Huang. End-to-end learning for lane keeping of self-driving cars. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1856–1860, 2017. Online: https://ieeexplore.ieee.org/document/7995975.

Aaron M. Cohen and William R. Hersh. A survey of current work in biomedical text mining. *Briefings in Bioinformatics*, 6(1):57–71, 2005. Online: https://dmice.ohsu.edu/hersh/briefings-05-cohen.pdf.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960. Online: https://journals.sagepub.com/doi/pdf/10.1177/001316446002000104.

Nigel Collier and Jin-Dong Kim. Introduction to the Bio-entity Recognition Task at JNLPBA. In Nigel Collier, Patrick Ruch, and Adeline Nazarenko, editors, *COLING 2004 International Joint workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP) 2004*, pages 73–78, Geneva, Switzerland, 2004. Online:http://aclweb.org/anthology/W04-1213.

Michael Connor and Dan Roth. Context Sensitive Paraphrasing with a Single Unsupervised Classifier. In *18th European Conference on Machine Learning (ECML)*, pages 289–295, Warsaw, Poland, 2007. Online: http://bit.do/contextsensitiveparaphrase.

William Coster and David Kauchak. Simple English Wikipedia: A New Text Simplification Task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, pages 665–669, Portland, OR, USA, 2011. Online: http://www.aclweb.org/anthology/P11-2117.

Koby Crammer and Yoram Singer. Ultraconservative Online Algorithms for Multiclass Problems. *Journal of Machine Learning Research*, 3:951–991, 2003. Online: http://www.jmlr.org/papers/volume3/crammer03a/crammer03a.pdf.

Kevin Crowston, Xiaozhong Liu, and Eileen E. Allen. Machine Learning and Rule-based Automated Coding of Qualitative Data. In *Proceedings of the 73rd ASIS&T Annual Meeting on Navigating Streams in an Information Ecosystem - Volume 47*, pages 108:1–108:2, Silver Springs, MD, USA, 2010. Online: https://onlinelibrary.wiley.com/doi/pdf/10.1002/meet.14504701328.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: An Architecture for Development of Robust HLT Applications. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 168–175, Philadelphia, PA, USA, 2002. Online: http://aclweb.org/anthology/P02-1022.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. *Text Processing with GATE (Version 6)*. 2011. Online: https://gate.ac.uk/releases/gate-6.1-build3913-ALL/tao.pdf.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. TiMBL: Tilburg Memory-Based Learner - version 1.0 - Reference Guide, 1998. Online: https://ilk.uvt.nl/downloads/pub/papers/Timbl_6.3_Manual.pdf.

Omid E. David, Nathan S. Netanyahu, and Lior Wolf. DeepChess: End-to-End Deep Neural Network for Automatic Learning in Chess. In Alessandro E.P. Villa, Paolo Masulli, and Antonio Javier Pons Rivero, editors, *Artificial Neural Networks and Machine Learning – ICANN 2016*, pages 88–96, 2016. ISBN 978-3-319-44781-0. Online: https://arxiv.org/pdf/1711.09667.pdf.

Elnaz Davoodi and Leila Kosseim. CLaC at SemEval-2016 Task 11: Exploring linguistic and psycho-linguistic Features for Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 982–985, San Diego, CA, USA, 2016. Online: http://aclweb.org/anthology/S16-1151.

Jan De Belder and Marie-Francine Moens. Text simplification for children. In *Proceedings of the SIGIR workshop on accessible search systems*, pages 19–26, Geneva, Switzerland, 2010. Online: https://core.ac.uk/download/pdf/34476855.pdf.

Dirk De Hertog and Anaïs Tack. Deep Learning Architecture for Complex Word Identification. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, LA, USA, 2018. Online: http://aclweb.org/anthology/W18-0539.

Pinar Donmez, Krysta M. Svore, and Christopher J.C. Burges. On the local optimality of lambdarank. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 460–467, Boston, MA, USA, 2009. Online: https://dl.acm.org/citation.cfm?id=1571941.1572021.

Steven M. Drucker, Danyel Fisher, and Sumit Basu. Helping Users Sort Faster with Adaptive Machine Learning Recommendations. In Pedro Campos, Nicholas Graham, Joaquim Jorge, Nuno Nunes, Philippe Palanque, and Marco Winckler, editors, *Human-Computer Interaction – INTERACT 2011*, pages 187–203, Berlin, Heidelberg, 2011. Online: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/icluster_interact.pdf.

Richard Eckart de Castilho, Chris Biemann, Iryna Gurevych, and **Seid Muhie Yimam**. WebAnno: a flexible, web-based annotation tool for CLARIN. In *CLARIN Annual Conference*, Soesterberg, The Netherlands, 2014. Online: https://www.clarin.eu/sites/default/files/cac2014_submission_6_0.pdf.

Richard Eckart de Castilho, Éva Mújdricza-Maydt, **Seid Muhie Yimam**, Silvana Hartmann, Iryna Gurevych, Anette Frank, and Chris Biemann. A web-based tool for the integrated annotation of semantic and syntactic structures. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 76–84, Osaka, Japan, 2016. Online: http://aclweb.org/anthology/W16-4011.

Lijun Feng, Noémie Elhadad, and Matt Huenerfauth. Cognitively motivated features for readability assessment. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 229–237, Athens, Greece, 2009. Online: https://dl.acm.org/citation.cfm?id=1609092.

David Ferrucci and Adam Lally. UIMA: An architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4): 327–348, 2004. Online: https://dl.acm.org/citation.cfm?id=1030325.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. Building Watson: An Overview of theDeepQA Project. pages 59–79, 2010. Online: https://aaai.org/ojs/index.php/aimagazine/article/view/2303.

Mark Alan Finlayson and Nidhi Kulkarni. Detecting Multi-word Expressions Improves Word Sense Disambiguation. In *Proceedings of the Workshop on Multiword Expressions: From Parsing and Generation to the Real World*, pages 20–24, Portland, OR, USA, 2011. Online: http://www.aclweb.org/anthology/W11-0805.

Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971. Online: http://psycnet.apa.org/buy/1972-05083-001.

Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. Automatic recognition of multi-word terms: the C-value/NC-value method. *International Journal on Digital Libraries*, 3(2):115–130, 2000. Online: https://link.springer.com/article/10.1007/s007999900023.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4:933–969, 2003. Online: https://dl.acm.org/citation.cfm?id=964285.

Geert Freyhoff, Gerhard Hess, Linda Kerr, Bror Tronbacke, and Kathy Van Der Veken. *Make it Simple, European Guidelines for the Production of Easy-toRead Information for People with Learning Disability*. ILSMH European Association, Brussels, Belgium, 1998. Online: http://bit.do/makeitsimple.

Binyam Gebrekidan Gebre. Part-of-Speech Tagging for Amharic. In *ISMTCL Proceedings, International Review Bulag, PUFC*, 2009. ISBN 978-2-84867-261-8. Online: http://clara.b.uib.no/files/2011/08/paper_for_bulag_binyam.pdf.

Omid Ghiasvand and Rohit Kate. UWM: Disorder mention extraction from clinical text using CRFs and normalization using learned edit distance patterns. In *Proceedings of SemEval 2014*, Dublin, Ireland, 2014. Online: http://www.aclweb.org/anthology/S14-2147.

Goran Glavaš and Sanja Štajner. Event-centered simplification of news stories. In *Proceedings of the Student Research Workshop at the International Conference on Recent Advances in Natural Language Processing*, pages 71–78, Hissar, Bulgaria, 2013. Online: http://www.aclweb.org/anthology/R13-2011.

Goran Glavaš and Sanja Štajner. Simplifying Lexical Simplification: Do We Need Simplified Corpora? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 63–68, Beijing, China, 2015. Online: http://www.aclweb.org/anthology/P15-2011.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. 2016. Online: http://www.deeplearningbook.org.

Sian Gooding and Ekaterina Kochmar. CAMB at CWI Shared Task 2018: Complex Word Identification with Ensemble-Based Voting. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, LA, USA, 2018. Online: http://aclweb.org/anthology/W18-0520.

David Graff. The AQUAINT Corpus of English News Text LDC2002T31. In *Web Download. Philadelphia: Linguistic Data Consortium*, 2002. Online: https://catalog.ldc.upenn.edu/LDC2002T31.

Zhou GuoDong and Su Jian. Exploring Deep Knowledge Resources in Biomedical Name Recognition. In *Proceedings of NLPBA/BioNLP at COLING'04*, pages 99–102, Geneva, Switzerland, 2004. Online: http://anthology.aclweb.org/W/W04/W04-1219.pdf.

Nathan Hartmann and Leandro Borges dos Santos. NILC at CWI 2018: Exploring Feature Engineering and Feature Learning. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, LA, USA, 2018. Online: http://aclweb.org/anthology/W18-0540.

Erhard W. Hinrichs, Marie Hinrichs, and Thomas Zastrow. WebLicht: Web-Based LRT Services for German. In *Proceedings of the ACL 2010 System Demonstrations*, pages 25–29, Uppsala, Sweden, 2010. Online: http://www.aclweb.org/anthology/P10-4005.

Gerold Hintz and Chris Biemann. Language transfer learning for supervised lexical substitution. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 118–129, Berlin, Germany, 2016. Online: http://www.aclweb.org/anthology/P16-1012.

David Hirsh and Paul Nation. What vocabulary size is needed to read unsimplified texts for pleasure? *Reading in a Foreign Language*, 8(2):689–696, 1992.

ChukFong Ho, Masrah Azrifah Azmi Murad, Shyamala Doraisamy, and Rabiah Abdul Kadir. Extracting lexical and phrasal paraphrases: a review of the literature. *Artificial Intelligence Review*, 42(4):851–894, 2014. Online: https://link.springer.com/article/10.1007/s10462-012-9357-8.

Thomas Ryan Hoens and Nitesh V. Chawla. Learning in Non-stationary Environments with Class Imbalance. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–176, Beijing, China, 2012. ISBN 978-1-4503-1462-6. Online: http://doi.acm.org/10.1145/2339530.2339558.

Andreas Holzinger. Human-Computer Interaction and Knowledge Discovery (HCI-KDD): What Is the Benefit of Bringing Those Two Fields to Work Together? In Alfredo Cuzzocrea, Christian Kittl, Dimitris E. Simos, Edgar Weippl, and Lida Xu, editors, *Availability, Reliability, and Security in Information Systems and HCI*, pages 319–328. Berlin, Heidelberg, 2013. Online: https://link.springer.com/chapter/10.1007/978-3-642-40511-2_22.

Andreas Holzinger. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics*, 3(2):119–131, 2016. URL https://doi.org/10.1007/s40708-016-0042-6. Online: https://link.springer.com/article/10.1007/s40708-016-0042-6.

Andreas Holzinger, Regina Geierhofer, Felix Modritscher, and Roland Tatzl. Semantic information in medical information systems: Utilization of text mining techniques to analyze medical diagnoses. *Journal of Universal Computer Science*, 14(22):3781–3795, 2008. Online: http://www.jucs.org/jucs_14_22/semantic_information_in_medical.

Andreas Holzinger, Pinar Yildirim, Michael Geier, and Klaus-Martin Simonic. Quality-Based Knowledge Discovery from Medical Text on the Web. In Gabriella Pasi, Gloria Bordogna, and Lakhmi C. Jain, editors, *Quality Issues in the Management of Web Information*, pages 145–158. Berlin, Heidelberg, 2013. ISBN 978-3-642-37688-7. Online: https://link.springer.com/chapter/10.1007/978-3-642-37688-7_7.

Andreas Holzinger, Johannes Schantl, Miriam Schroettner, Christin Seifert, and Karin Verspoor. Biomedical Text Mining: State-of-the-Art, Open Problems and Future Challenges. In Andreas Holzinger and Igor Jurisica, editors, *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics, LNCS 8401*, pages 271–300. 2014. Online: https://link.springer.com/chapter/10.1007/978-3-662-43968-5_16.

Colby Horn, Cathryn Manduca, and David Kauchak. Learning a Lexical Simplifier Using Wikipedia. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 458–463, Baltimore, MD, USA, 2014. Online: http://www.aclweb.org/anthology/P14-2075.

Eduard H. Hovy and Julia Lavid. Towards a 'Science' of Corpus Annotation: A New Methodological Challenge for Corpus Linguistics. *International Journal of Translation Studies*, 22 (1), 2013. Online: https://www.cs.cmu.edu/~hovy/papers/10KNS-annotation-Hovy-Lavid.pdf.

Nancy Ide and Keith Suderman. GrAF: A Graph-based Format for Linguistic Annotations. In *Proceedings of the Linguistic Annotation Workshop*, pages 1–8, Prague, Czech Republic, 2007. Online: https://www.cs.vassar.edu/~ide/papers/LAW.pdf.

Nancy Ide and Keith Suderman. The Linguistic Annotation Framework: A Standard for Annotation Interchange and Merging. *Language Resources and Evaluation*, 48(3):395–418, 2014. ISSN 1574-020X. Online: https://link.springer.com/article/10.1007/s10579-014-9268-1.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information and System Security*, 20(4):422–446, 2002. ISSN 1046-8188. Online: http://doi.acm.org/10.1145/582415.582418.

Tomoyuki Kajiwara and Mamoru Komachi. Complex Word Identification Based on Frequency in a Learner Corpus . In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, LA, USA, 2018. Online: http://aclweb.org/anthology/W18-0521.

David Kauchak. Improving Text Simplification Language Modeling Using Unsimplified Text Data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1537–1546, Sofia, Bulgaria, 2013. Online: http://www.aclweb.org/anthology/P13-1151.

David Kauchak and Regina Barzilay. Paraphrasing for Automatic Evaluation. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 455–462, New York, NY, USA, 2006. Online: http://aclweb.org/anthology/N06-1058.

Esther Kaufmann and Abraham Bernstein. How Useful Are Natural Language Interfaces to the Semantic Web for Casual End-users? In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, pages 281–294, Busan, South Korea, 2007. ISBN 3-540-76297-3, 978-3-540-76297-3. Online: https://link.springer.com/chapter/10.1007/978-3-540-76298-0_21.

Parambir S. Keila and David B. Skillicorn. Structure in the Enron Email Dataset. *Computational and Mathematical Organization Theory*, 11(3):183–199, 2005. Online: https://link.springer.com/article/10.1007/s10588-005-5379-y.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. Overview of BioNLP'09 Shared Task on Event Extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 1–9, Boulder, CO, USA, 2009. Online: http://aclweb.org/anthology/W09-1401.

Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, Ngan Nguyen, and Jun'ichi Tsujii. Overview of BioNLP Shared Task 2011. In *Proceedings of BioNLP*, pages 1–6, Portland, OR, USA, 2011. Online: https://www.aclweb.org/anthology/W/W11/W11-1801.pdf.

David Klaper, Sarah Ebling, and Martin Volk. Building a German/Simple German Parallel Corpus for Automatic Text Simplification. In *Proceedings of the Second Workshop on Predicting and*

Improving Text Readability for Target Reader Populations, pages 11–19, Sofia, Bulgaria, 2013. Online: http://aclweb.org/anthology/W13-2902.

Reinhard Kneser and Hermann Ney. Improved backing-off for M-gram language modeling. In *The 1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184 vol.1, Detroit, MI, USA, 1995. Online: https://ieeexplore.ieee.org/document/479394.

Zornitsa Kozareva and Andrés Montoyo. Paraphrase Identification on the Basis of Supervised Machine Learning Techniques. In Tapio Salakoski, Filip Ginter, Sampo Pyysalo, and Tapio Pahikkala, editors, *Advances in Natural Language Processing*, pages 524–533, Turku, Finland, 2006. Online: https://link.springer.com/chapter/10.1007/11816508_52.

Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. What Substitutes Tell Us - Analysis of an "All-Words" Lexical Substitution Corpus. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 540–549, Gothenburg, Sweden, 2014. Online: http://www.aclweb.org/anthology/E14-1057.

Klaus Krippendorff. Computing Krippendorff's Alpha-Reliability. 2011. Online: http://repository.upenn.edu/asc_papers/43.

Todd Kulesza, Saleema Amershi, Rich Caruana, Danyel Fisher, and Denis Charles. Structured Labeling for Facilitating Concept Evolution in Machine Learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3075–3084, Toronto, ON, Canada, 2014. ISBN 978-1-4503-2473-1. Online: http://doi.acm.org/10.1145/2556288.2557238.

Walter S. Lasecki, Luz Rello, and Jeffrey P. Bigham. Measuring text simplification with the crowd. In *Proceedings of the 12th Web for All Conference*, pages 4:1–4:9, Florence, Italy, 2015. Online: https://dl.acm.org/citation.cfm?id=2746658.

Chih Lee, Wen-Juan Hou, and Hsin-Hsi Chen. Annotating Multiple Types of Biomedical Entities: A Single Word Classification Approach. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, pages 80–83, Geneva, Switzerland, 2004. Online: http://aclweb.org/anthology/W04-1215.

Ulf Leser and Jörg Hakenberg. What makes a gene name? Named entity recognition in the biomedical literature. *Briefings in Bioinformatics*, 6(4):357–69, 2005. Online: https://academic.oup.com/bib/article-pdf/1/4/357/7726594/357.pdf.

Abby Levenberg, Chris Callison-Burch, and Miles Osborne. Stream-based Translation Models for Statistical Machine Translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 394–402, Los Angeles, CA, USA, 2010. Online: http://aclweb.org/anthology/N10-1062.

Hang Li. *Learning to Rank for Information Retrieval and Natural Language Processing: Second Edition*. 2nd edition, 2014. ISBN 1627055843, 9781627055840.

Ping Li, Christopher J. C. Burges, and Qiang Wu. McRank: Learning to Rank Using Multiple Classification and Gradient Boosting. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 897–904, Vancouver, BC, Canada, 2007. ISBN 978-1-60560-352-0. Online: http://dl.acm.org/citation.cfm?id=2981562.2981675.

Franklin M. Liang. *Word Hy-phen-a-tion by Com-put-er*. PhD thesis, Stanford University, Department of Linguistics, Stanford, CA., USA, 1983. Online: https://tug.org/docs/liang/.

Todd Lingren, Louise Deleger, Katalin Molnar, Haijun Zhaiand Jareen Meinzen-Derr, Megan Kaiser, Laura Stoutenborough, Qi Li, and Imre Solti. Evaluating the impact of pre-annotation on annotation speed and potential bias: natural language processing gold standard development for clinical named entity recognition in clinical trial announcements. *Journal of the American Medical Informatics Association*, pages 951 – 991, 2013. Online: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3994857/.

Marcus-Christopher Ludl, Achim Lewandowski, and Georg Dorffner. Adaptive Machine Learning in Delayed Feedback Domains by Selective Relearning. *Applied Artificial Intelligence*, 22(6):543–557, 2008.

Xiaoyi Ma, Haejoong Lee, Steven Bird, and Kazuaki Maeda. Models and Tools for Collaborative Annotation. In *LREC 2002, Third International Conference on Language Resources and Evaluation*, pages 2066 – 2073, Las Palmas, Canary Islands, Spain, 2002. Online: http://lrec.elra.info/proceedings/lrec2002/.

Diana McCarthy and Roberto Navigli. SemEval-2007 Task 10: English Lexical Substitution Task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 48–53, Prague, Czech Republic, 2007. Online: http://aclweb.org/anthology/S07-1009.

Diana McCarthy, Bill Keller, and John Carroll. Detecting a Continuum of Compositionality in Phrasal Verbs. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, pages 73–80, Sapporo, Japan, 2003. Online: http://www.aclweb.org/anthology/W03-1810.

Oren Melamud, Omer Levy, and Ido Dagan. A simple word embedding model for lexical substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 1–7, Denver, CO, USA, 2015. Online: http://www.aclweb.org/anthology/W15-1501.

Mencap. Am I making myself clear? Mencap's guidelines for accessible writing. 2002. Online: http://www.accessibleinfo.co.uk/pdfs/Making-Myself-Clear.pdf.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*, pages 3111–3119, Stateline, NV, USA, 2013. Online: https://arxiv.org/abs/1310.4546.

George A. Miller. WordNet: A Lexical Database for English. *Commun. ACM*, 38(11):39–41, 1995. ISSN 0001-0782. Online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.1823&rep=rep1&type=pdf.

Thomas Morton and Jeremy LaCivita. WordFreak: an open tool for linguistic annotation. In *Companion Volume of the Proceedings of HLT-NAACL 2003 - Demonstrations*, pages 17–18, Edmonton, AB, Canada, 2003. Online: http://aclweb.org/anthology/N03-4009.

Christoph Müller and Michael Strube. Multi-level annotation of linguistic data with MMAX2. In S. Braun, K. Kohn, and J. Mukherjee, editors, *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*, pages 197–214. Frankfurt a.M., Germany, 2006. Online: http://mmax2.sourceforge.net/mmaxpaper.pdf.

Martin Müller, Kathrin Ballweg, Tatiana von Landesberger, **Seid Muhie Yimam**, Uli Fahrer, Chris Biemann, Marcel Rosenbach, Michaela Regneri, and Heiner Ulrich. Guidance for Multi-Type Entity Graphs from Text Collections. In Michael Sedlmair and Christian Tominski, editors, *EuroVis Workshop on Visual Analytics (EuroVA)*, Barcelona, Spain, 2017. ISBN 978-3-03868-042-0. Online: https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2017-mueller-eurova-newsleak-guidance.pdf.

Paul I. S. Nation. *Learning vocabulary in another language*. Cambridge: Cambridge University Press, 2001. Online: http://catdir.loc.gov/catdir/samples/cam031/2001269892.pdf.

Nils J. Nilsson. The Quest for Artificial Intelligence. Cambridge University Press, New York, NY, USA, 1st edition, 2009. ISBN 0521122937, 9780521122931. Online: https://ai.stanford.edu/~nilsson/QAI/qai.pdf.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 Shared Task on Dependency Parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic, 2007. Online: http://www.aclweb.org/anthology/D07-1096.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. The GENIA corpus: An annotated research abstract corpus in molecular biology domain. In *Proceedings of Human Language Technology Research*, pages 82–86, San Francisco, CA, USA, 2002. Online: https://dl.acm.org/citation.cfm?id=1289260.

Naoaki Okazaki. CRFsuite: a fast implementation of Conditional Random Fields (CRFs), 2007. Online: http://www.chokkan.org/software/crfsuite/.

Gustavo Paetzold and Lucia Specia. LEXenstein: A Framework for Lexical Simplification. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 85–90, Beijing, China, 2015. Online: http://aclweb.org/anthology/P15-4015.

Gustavo Paetzold and Lucia Specia. Unsupervised Lexical Simplification for Non-native Speakers. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3761–3767, Phoenix, AZ, USA, 2016a. Online: https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/download/12235/12158.

Gustavo Paetzold and Lucia Specia. Inferring Psycholinguistic Properties of Words. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 435–440, San Diego, CA, USA, June 2016b. Online: http://www.aclweb.org/anthology/N16-1050.

Gustavo Paetzold and Lucia Specia. SemEval 2016 Task 11: Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569, San Diego, CA, USA, 2016c. Online: http://www.aclweb.org/anthology/S16-1085.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual Lifelong Learning with Neural Networks: A Review. *ArXiv e-prints, 1802.07569*, 2018. Online: https://arxiv.org/abs/1802.07569.

Ellie Pavlick and Chris Callison-Burch. Simple PPDB: A Paraphrase Database for Simplification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 143–148, Berlin, Germany, 2016. Online:www.aclweb.org/anthology/P16-2024.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China, 2015. Online: http://aclweb.org/anthology/P15-2070.

Sarah E. Petersen and Mari Ostendorf. Text Simplification for Language Learners: A Corpus Analysis. In *Proceedings of Workshop on Speech and Language Technology for Education*, pages 69–72, Farmington, PA, USA, 2007. Online: https://www.isca-speech.org/archive_open/archive_papers/slate_2007/sle7_069.pdf.

Slav Petrov, Dipanjan Das, and Ryan McDonald. A Universal Part-of-Speech Tagset. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, 2012. Online: http://www.lrec-conf.org/proceedings/lrec2012/pdf/274_Paper.pdf.

Emanuele Pianta and Luisa Bentivogli. Annotating Discontinuous Structures in XML: the Multiword Case. In *In Proceedings of the LREC 2004 Satellite Workshop on*, pages 30–37, Lisbon, Portugal, 2004. Online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.1290&rep=rep1&type=pdf.

Ildikó Pilán, Elena Volodina, and Richard Johansson. Rule-based and machine learning approaches for second language sentence-level readability. In *Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 174–184, Baltimore, MD, USA, 2014. Online: http://www.aclweb.org/anthology/W14-1821.

PlainLanguage. Federal plain language guidelines, 2011. Online: https://plainlanguage.gov/media/FederalPLGuidelines.pdf.

Maja Popović. Complex Word Identification using Character n-grams. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, LA, USA, 2018. Online: http://aclweb.org/anthology/W18-0541.

James Pustejovsky and Amber Stubbs. *Natural Language Annotation for Machine Learning*. 2012. ISBN 978-1-4493-0666-3.

Sampo Pyysalo, Tomoko Ohta, and Jun'ichi Tsujii. Overview of the Entity Relations (REL) Supporting Task of BioNLP Shared Task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 83–88, Portland, OR, USA, 2011. Online: https://dl.acm.org/citation.cfm?id=2107703.

Carlos Ramisch, Aline Villavicencio, and Christian Boitet. mwetoolkit: a Framework for Multiword Expression Identification. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, pages 134–136, Valletta, Malta, 2010. Online: http://www.lrec-conf.org/proceedings/lrec2010/pdf/803_Paper.pdf.

Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, 2010. Online: https://radimrehurek.com/gensim/lrec2010_final.pdf.

Luz Rello, Ricardo Baeza-Yates, Laura Dempere-Marco, and Horacio Saggion. Frequent words improve readability and short words improve understandability for people with dyslexia. In *Proceedings of the INTERACT 2013: 14th IFIP TC13 Conference on Human-Computer Interaction., 2013*, Cape Town, South Africa, 2013. Online: https://link.springer.com/chapter/10.1007/978-3-642-40498-6_15.

Thomas C. Rindflesch, Lorraine Tanabe, John N. Weinstein, and Lawrence Hunter. EDGAR: Extraction of Drugs, Genes And Relations from the Biomedical Literature. *Pacific Symposium on Biocomputing*, 2000. Online: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2709525/.

Barbara Rosario and Marti A. Hearst. Multi-way Relation Classification: Application to Protein-protein Interactions. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 732–739, Vancouver, BC, Canada, 2005. Online: http://aclweb.org/anthology/H05-1092.

Eugen Ruppert, Manuel Kaufmann, Martin Riedl, and Chris Biemann. JOBIMVIZ: A Web-based Visualization for Graph-based Distributional Semantic Models. In *The Annual Meeting of the Association for Computational Linguistics (ACL) System Demonstrations*, pages 103–108, Beijing, China, 2015. Online: http://www.aclweb.org/anthology/P15-4018.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann A. Copestake, and Dan Flickinger. Multiword Expressions: A Pain in the Neck for NLP. In *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, pages 1–15, London, UK, 2002. ISBN 3-540-43219-1. Online: http://lingo.stanford.edu/pubs/WP-2001-03.pdf.

Horacio Saggion, Sanja Štajner, Stefan Bott, Simon Mille, Luz Rello, and Biljana Drndarevic. Making It Simplext: Implementation and Evaluation of a Text Simplification System for Spanish. *ACM Transactions on Accessible Computing (TACCESS)*, 6(4):14:1–14:36, 2015. ISSN 1936-7228. Online: https://dl.acm.org/citation.cfm?id=2738046.

Guido Sautter, Klemens Böhm, Frank Padberg, and Walter Tichy. Empirical Evaluation of Semi-automated XML Annotation of Text Documents with the GoldenGATE Editor. In László Kovács, Norbert Fuhr, and Carlo Meghini, editors, *Research and Advanced Technology for Digital Libraries*, Budapest, Hungary, 2007. ISBN 978-3-540-74851-9. Online: https://link.springer.com/chapter/10.1007/978-3-540-74851-9_30.

Nathan Schneider and Noah A. Smith. A Corpus and Model Integrating Multiword Expressions and Supersenses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1537–1547, Denver, CO, USA, 2015. Online: www.aclweb.org/anthology/N15-1177.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah Smith. Discriminative Lexical Semantic Segmentation with Gaps: Running the MWE Gamut. *Transactions of the Association for Computational Linguistics*, 2:193–206, 2014. Online: http://aclweb.org/anthology/Q14-1016.

Burr Settles. Active learning literature survey. Technical report, University of Wisconsin–Madison, 2010. Online: http://burrsettles.com/pub/settles.activelearning.pdf.

Matthew Shardlow. The CW Corpus: A New Resource for Evaluating the Identification of Complex Words. In *Proceedings of the Second Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pages 69–77, Sofia, Bulgaria, 2013. Online: aclweb.org/anthology/W13-2908.

Matthew Shardlow. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, 4(1), 2014. Online: https://www.researchgate.net/publication/270553446_A_Survey_of_Automated_Text_Simplification.

Advaith Siddharthan. Syntactic Simplification and Text Cohesion. *Research on Language and Computation*, 4(1):77–109, 2006. ISSN 1572-8706. Online: http://www.cs.columbia.edu/nlp/papers/2004/siddharthan_04.pdf.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander

Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. BRAT: a Web-based Tool for NLP-Assisted Text Annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France, 2012. Online: http://aclweb.org/anthology/E12-2021.

Simone Stumpf, Vidya Rajaram, Lida Li, Margaret Burnett, Thomas Dietterich, Erin Sullivan, Russell Drummond, and Jonathan Herlocker. Toward Harnessing User Feedback for Machine Learning. In *Proceedings of the 12th international conference on Intelligent user interfaces*, pages 82–91, Honolulu, HI, USA, 2007. Online: https://dl.acm.org/citation.cfm?id=1216316.

Julia Suter, Sarah Ebling, and Martin Volk. Rule-based Automatic Text Simplification for German. In *Proceedings of the 13th Conference on Natural Language Processing (KONVENS 2016)*, pages 279–287, Bochum, Germany, 2016. Online: https://www.linguistics.rub.de/konvens16/pub/35_konvensproc.pdf.

György Szarvas, Róbert Busa-Fekete, and Eyke Hüllermeier. Learning to Rank Lexical Substitutions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*, pages 1926–1932, Seattle, WA, USA, 2013. Online: http://www.aclweb.org/anthology/D13-1198.

Valentin Tablan, Kalina Bontcheva, Diana Maynard, and Hamish Cunningham. Ollie: Online learning for information extraction. In Jon Patrick and Hamish Cunningham, editors, *Proceedings of the HLT-NAACL 2003 Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS)*, Edmonton, AB, Canada, 2003. Online: http://www.aclweb.org/anthology/W03-0803.

Anaïs Tack, Thomas François, Anne-Laure Ligozat, and Cédrick Fairon. Evaluating Lexical Simplification and Vocabulary Knowledge for Learners of French: Possibilities of Using the FLELex Resource. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 230–236, Portorož, Slovenia, 2016. Online: http://www.lrec-conf.org/proceedings/lrec2016/pdf/544_Paper.pdf.

Yuka Tateisi and Jun'ichi Tsujii. Part-of-Speech Annotation of Biology Research Abstracts. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*

(LREC'04), pages 1267–1270, Lisbon, Portugal, 2004. Online: http://www.lrec-conf.org/proceedings/lrec2004/pdf/528.pdf.

Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and Jun'ichi Tsujii. Syntax Annotation for the GENIA Corpus. In *Companion Volume to the Proceedings of Conference including Posters/Demos and tutorial abstracts*, pages 222–227, Lisbon, Portugal, 2005. Online: http://aclweb.org/anthology/I05-2038.

Stefan Thater, Georgiana Dinu, and Manfred Pinkal. Ranking Paraphrases in Context. In *Proceedings of the 2009 Workshop on Applied Textual Inference*, pages 44–47, Suntec, Singapore, 2009. Online: https://dl.acm.org/citation.cfm?id=1708149.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *North American Chapter of the Association for Computational Linguistics – Human Language Technologies (NAACL HLT 2003)*, pages 982–985, Edmonton, AB, Canada, 2003. Online: http://aclweb.org/anthology/N03-1033.

Jun'ichi Tsujii. Computational Linguistics and Natural Language Processing. In *Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I*, pages 52–67, Tokyo, Japan, 2011. ISBN 978-3-642-19399-6. Online: https://link.springer.com/chapter/10.1007/978-3-642-19400-9_5.

Yulia Tsvetkov and Shuly Wintner. Extraction of Multi-word Expressions from Small Parallel Corpora. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1256–1264, Beijing, China, 2010. Online: https://dl.acm.org/citation.cfm?id=1944710.

Alexey Tsymbal. The Problem of Concept Drift: Definitions and Related Work. Technical report, Department of Computer Science, Trinity College: Dublin, Ireland, 2004. Online: https://www.scss.tcd.ie/publications/tech-reports/reports.04/TCD-CS-2004-15.pdf.

Özlem Uzuner, Yuan Luo, and Peter Szolovits. Evaluating the State-of-the-Art in Automatic De-identification. *Journal of the American Medical Informatics Association*, 14(5):550–563, 2007. Online: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1975792/.

Özlem Uzuner, Imre Solti, Fei Xia, and Eithon Cadag. Community annotation experiment for ground truth generation for the i2b2 medication challenge. *Journal of the American Medical Informatics Association*, 17(5):561–570, 2010. Online: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2995684/.

Kees van Deemter and Rodger Kibble. What is Coreference, and What Should Coreference Annotation Be? In *Proceedings of the Workshop on Coreference and Its Applications*, pages 90–96, College Park, MD, USA, 1999. Online: http://www.aclweb.org/anthology/W99-0213.

Maarten van Gompel and Martin Reynaert. FoLiA: A practical XML format for linguistic annotation - a descriptive and comparative study. *Computational Linguistics in the Netherlands Journal*, 3:63–81, 2013. Online: https://www.clinjournal.org/node/39.

Horacio Vico and Daniel Calegari. Software architecture for document anonymization. *Electronic Notes in Theoretical Computer Science*, 314:83–100, 2015. ISSN 1571-0661. Online: https://www.sciencedirect.com/science/article/pii/S1571066115000298.

Vedrana Vidulin, Marko Bohanec, and Matjaž Gams. Combining human analysis and machine data mining to obtain credible data relations. *Information Sciences*, 288:254–278, 2014. Online: https://www.sciencedirect.com/science/article/pii/S0020025514008032.

Veronika Vincze, István Nagy T., and Gábor Berend. Multiword Expressions and Named Entities in the Wiki50 Corpus. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 289–295, Hissar, Bulgaria, 2011. Online: http://aclweb.org/anthology/R11-1040.

Sanja Štajner, Marc Franco-Salvador, Simone Paolo Ponzetto, Paolo Rosso, and Heiner Stuckenschmidt. Sentence Alignment Methods for Improving Text Simplification Systems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 97–102, Vancouver, BC, Canada, 2017. Online:http://aclweb.org/anthology/P17-2016.

Jialei Wang, Ji Wan, Yongdong Zhang, and Steven Hoi. SOLAR: Scalable Online Learning Algorithms for Ranking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1692–1701, Beijing, China, 2015. Online: http://www.aclweb.org/anthology/P15-1163.

Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Wei Chen, and Tie-Yan Liu. A Theoretical Analysis of Normalized Discounted Cumulative Gain (NDCG) Ranking Measures. In *Proceedings of the 26th Annual Conference on Learning Theory (COLT 2013)*, pages 25–54, Princeton, NJ, USA, 2013. Online: http://proceedings.mlr.press/v30/Wang13.pdf.

Nikhil Wani, Sandeep Mathias, Jayashree Aanand Gajjam, and Pushpak Bhattacharyya. The Whole is Greater than the Sum of its Parts: Towards the Effectiveness of Voting Ensemble Classifiers for Complex Word Identification. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, LA, USA, 2018. Online: http://aclweb.org/anthology/W18-0522.

Gregor Wiedemann, **Seid Muhie Yimam**, and Chris Biemann. A Multilingual Information Extraction Pipeline for Investigative Journalism. In *Proceedings of 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, Brussels, Belgium, 2018a. Online: http://aclweb.org/anthology/D18-2014.

Gregor Wiedemann, **Seid Muhie Yimam**, and Chris Biemann. New/s/leak 2.0 – Multilingual Information Extraction and Visualization for Investigative Journalism. In *Proceedings of the 10th International Conference on Social Informatics (SocInfo 2018)*, St.Petersburg, Russia, 2018b. Online: https://arxiv.org/abs/1807.05151.

Krzysztof Wróbel. PLUJAGH at SemEval-2016 Task 11: Simple System for Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 953–957, San Diego, CA, USA, 2016. Online: http://www.aclweb.org/anthology/S16-1146.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199, New York, NY, USA, 2008. Online: http://icml2008.cs.helsinki.fi/papers/167.pdf.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3: 283–297, 2015. Online: https://cocoxu.github.io/publications/tacl2015-text-simplification-opinion.pdf.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. Optimizing Statistical Machine Translation for Text Simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415, 2016. Online: http://cs.jhu.edu/~napoles/res/tacl2016-optimizing.pdf.

Meliha Yetisgen-Yildiz, Imre Solti, Fei Xia, and Scott Russell Halgrim. Preliminary experience with amazon's mechanical turk for annotating medical named entities. In *Proceedings of NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 180–183, Los Angeles, CA, USA, 2010. Online:aclweb.org/anthology/W10-0728.

Seid Muhie Yimam. Narrowing the Loop: Integration of Resources and Linguistic Dataset Development with Interactive Machine Learning. In *Proceedings of HLT-NAACL: Student Research Workshop*, pages 88–95, Denver, CO, USA, 2015. Online: http://www.aclweb.org/anthology/N15-2012.

Seid Muhie Yimam and Chris Biemann. Par4Sim – Adaptive Paraphrasing for Text Simplification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 331–342, Santa Fe, NM, USA, 2018. Online: http://aclweb.org/anthology/C18-1028.

Seid Muhie Yimam and Chris Biemann. Demonstrating PAR4SEM – A Semantic Writing Aid with Adaptive Paraphrasing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 48–53, Brussels, Belgium, 2018. Online: http://aclweb.org/anthology/D18-2009.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations. In *Proceedings of ACL 2013 System Demonstrations*, pages 1–6, Sofia, Bulgaria, 2013. Online: http://www.aclweb.org/anthology/P13-4001.pdf.

Seid Muhie Yimam, Richard Eckart de Castilho, Iryna Gurevych, and Chris Biemann. Automatic Annotation Suggestions and Custom Annotation Layers in WebAnno. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. System Demonstrations*, pages 91–96, Baltimore, MD, USA, 2014. Online: http://www.aclweb.org/anthology/P14-5016.

Seid Muhie Yimam, Chris Biemann, Ljiljana Majnarić, Šefket Šabanović, and Andreas Holzinger. Interactive and Iterative Annotation for Biomedical Entity Recognition. In Yike Guo, Karl Friston, Faisal Aldo, Sean Hill, and Hanchuan Peng, editors, *Brain Informatics and Health*, pages 347–357, London, UK, 2015. ISBN 978-3-319-23344-4. Online: https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2015-yimametal-bih-london.pdf.

Seid Muhie Yimam, Chris Biemann, Ljiljana Majnarić, Šefket Šabanović, and Andreas Holzinger. An adaptive annotation approach for biomedical entity and relation recognition. *Brain Informatics*, 3(3):157–168, 2016a. Online: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4999566/.

Seid Muhie Yimam, Héctor Martínez Alonso, Martin Riedl, and Chris Biemann. Learning Paraphrasing for Multiword Expressions. In *Proceedings of the 12th Workshop on Multiword*

Expressions, pages 1–10, Berlin, Germany, 2016b. Online: http://www.aclweb.org/anthology/W16-1801.

Seid Muhie Yimam, Heiner Ulrich, Tatiana von Landesberger, Marcel Rosenbach, Michaela Regneri, Alexander Panchenko, Franziska Lehmann, Uli Fahrer, Chris Biemann, and Kathrin Ballweg. New/s/leak – Information Extraction and Visualization for Investigative Data Journalists. In *Proceedings of ACL-2016 System Demonstrations*, pages 163–168, Berlin, Germany, 2016c. Online: http://www.aclweb.org/anthology/P16-4028.

Seid Muhie Yimam, Steffen Remus, Alexander Panchenko, Andreas Holzinger, and Chris Biemann. Entity-Centric Information Access with Human in the Loop for the Biomedical Domain. In *Proceedings of the Biomedical NLP Workshop associated with RANLP 2017*, pages 42–48, Varna, Bulgaria, 2017a. Online: https://doi.org/10.26615/978-954-452-044-1_006.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. Multilingual and Cross-Lingual Complex Word Identification. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 813–822, Varna, Bulgaria, 2017b. Online: https://doi.org/10.26615/978-954-452-049-6_104.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. CWIG3G2 - Complex Word Identification Task across Three Text Genres and Two User Groups. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 401–407, Taipei, Taiwan, 2017c. Online: http://www.aclweb.org/anthology/I17-2068.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. A Report on the Complex Word Identification Shared Task 2018. In *Proceedings of The 13th Workshop on Innovative Use of NLP for Building Educational Applications, NAACL 2018 Workshops*, pages 66–78, New Orleans, LA, USA, 2018. Online: http://aclweb.org/anthology/W18-0507.

Shiqi Zhao, Ting Liu, Xincheng Yuan, Sheng Li, and Yu Zhang. Automatic Acquisition of Context-Specific Lexical Paraphrases. In *International Joint Conference on Artificial Intelligence*, pages 1789–1794, Hyderabad, India, 2007. Online: http://www.aaai.org/Papers/IJCAI/2007/IJCAI07-289.pdf.

Indrė Žliobaitė, Mykola Pechenizkiy, and João Gama. An overview of concept drift applications. In Nathalie Japkowicz and Jerzy Stefanowski, editors, *Big Data Analysis: New Algorithms for a New Society*, pages 91–114. 2016. Online: https://www.win.tue.nl/~mpechen/publications/pubs/CD_applications15.pdf.