Wahed Hemati
3007 80 822 147
Computer science
Text Technology Lab
hemati@em.uni-frankfurt.de

Dissertation

TextImager-VSD

Large Scale Verb Sense Disambiguation and Named Entity Recognition in the Context of TextImager

Abdul Wahed Hemati

Frankfurt, 12.2019

Goethe University Frankfurt am Main
Prof. Alexander Mehler
Prof. Visvanathan Ramesh

1. Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Dissertation unterstützt und motiviert haben. Die Arbeit wäre ohne Unterstützung einer Vielzahl von Leuten nicht möglich gewesen.

Ganz besonders gilt dieser Dank Herrn Prof. Dr. Alexander Mehler, der meine Arbeit und somit auch mich betreut hat. Durch stetig kritisches Hinterfragen und konstruktive Kritik gab er mir wertvolle Hinweise mit auf den Weg. Er stand nicht nur mit wissenschaftlichem Rat und fachlichen Diskussionen zur Seite, sondern auch mit moralischer Unterstützung und Motivation. Die zahlreichen Gespräche auf intellektueller und persönlicher Ebene werden mir immer als bereichernder und konstruktiver Austausch in Erinnerung bleiben. Ich bin dankbar dafür, dass er mich stets sehr gut betreut und dazu gebracht hat, über meine Grenzen hinaus zu denken. Vielen Dank für die Geduld und Mühen.

Ich danke Herrn Prof. Dr. Visvanathan Ramesh für die hilfsbereite und wissenschaftliche Betreuung als Zweitgutachter.

Ich bedanke mich für die Möglichkeit, diese Arbeit im Text-Technology-Lab der Goethe-Universität Frankfurt anfertigen zu können. Ich hatte die Möglichkeit, von fachlich kompetenten Kollegen und Kolleginnen beraten und betreut zu werden. Dadurch bin ich fachlich und zwischenmenschlich mit ihnen gewachsen.

Nicht zuletzt gebührt meinen Eltern der höchste Dank, die während des Lebens und vor allem während des Studiums immer an meiner Seite standen. Sie haben mich in jeglicher Hinsicht unterstützt und motiviert. Danken möchte ich außerdem meiner Familie und meinen Freunden, die mich während der Zeit begleitet haben.

Acknowledgement

I would like to take this opportunity to thank all those who supported and motivated me during the writing of this dissertation. The work would not have been possible without the support and advice of many people.

I would especially like to thank Prof. Dr. Alexander Mehler, who supervised my work and thus also me. By constantly critical questioning and constructive criticism he gave me valuable hints on the way. He was not only there with scientific advice and technical discussions, but also with moral encouragement and motivation. The numerous discussions on an intellectual and personal level will always remain in my memory as an enriching and constructive exchange. I am grateful that he has always supported me and pushed me to go beyond my borders. Thank you very much for your patience and efforts.

I would like to thank Prof. Dr. Visvanathan Ramesh for his helpful and scientific support as second supervisor.

I am grateful for the opportunity of having done this work at the Text Technology Lab at the Goethe University in Frankfurt. I had the opportunity to be advised and supervised by competent colleagues. This has enabled me to grow both professionally and interpersonally with them.

Last but not least, I would like to thank my parents, who have always been by my side throughout my life and especially during my studies. They supported and motivated me in every respect. I would also like to thank my family and friends who accompanied me during this time.

2. Abstract

Durch das World Wide Web steigt die Zahl der frei zugänglichen Textdaten, was zu einem zunehmenden Interesse an der Forschung im Bereich der Computerlinguistik (CL) geführt hat. Dieser Bereich beschäftigt sich mit theoretisch orientierter Forschung zur Beantwortung der Frage, auf welche Art Sprache und Wissen repräsentiert werden muss, um Sprache verstehen und produzieren zu können. Dafür werden mathematische Modelle entwickelt, um die Phänomene in den verschiedenen Ebenen in menschlichen Sprachen zu erfassen. Ein weiteres Forschungsgebiet erfährt einen Interessenzuwachs, das eng mit CL zusammenhängt, nämlich das Natural Language Processing (NLP), bei dem es primär darum geht, effektive und effiziente Datenstrukturen und Algorithmen zu entwickeln, welche die mathematischen Modelle der CL implementieren.

Mit zunehmendem Interesse an diesen Bereichen werden in kürzeren Veröffentlichungszyklen NLP-Werkzeuge entwickelt und durch den *Open Source Trend* frei zur Verfügung gestellt, die verschiedene CL-Modelle implementieren, mit denen unterschiedliche Ebenen der Sprache verarbeitet werden können. Aufgrund des noch nicht definierten I/O-Standards für NLP führt das schnelle Wachstum jedoch zu einer heterogenen NLP-Landschaft, in der die Spezialisierungen der Tools wegen der Schnittstelleninkompatibilität nicht voneinander profitieren können. Daneben erfordert die ständig wachsende Menge an frei zugänglichen Textdaten eine performante Verarbeitungslösung. Diese Performanz kann durch horizontale und vertikale Skalierung von Hard- und Software Komponenten erreicht werden.

Aus diesen Gründen beschäftigt sich die vorliegende Arbeit im ersten Teil mit der Homogenisierung der NLP-Toollandschaft, welche durch ein standardisiertes Framework erzielt wird, das - `TextImager` - benannt wurde. Es stellt ein Cloud

Computing basiertes Multi-Service, -Server, -Pipeline, -Database, -User, -Repräsentations und -Visualisierungs Framework dar, das bereits eine Vielzahl an Werkzeugen für verschiedene Sprachen zur Verfügung stellt, mit denen verschiedene Ebenen der Sprache verarbeitet werden können. Damit ist es möglich, Forschungsfragen zu beantworten, die es erfordern, eine große Anzahl an Daten auf mehreren linguistischen Ebenen zu verarbeiten.

Durch die integrierten Werkzeuge und die homogenisierten I/O-Datenströme des TextImagers ist es nun möglich, die eingebauten Werkzeuge auf zwei Dimensionen zu kombinieren: (1) die horizontale Dimension, um eine NLP aufgabenspezifische Verbesserung zu erreichen (2) die orthogonale Dimension, um CL Modelle implementieren zu können, die auf mehrere Ebenen der Sprache aufbauen und somit auf eine Kombination aus unterschiedlichen NLP Werkzeugen angewiesen sind.

Der zweite Teil dieser Arbeit beschäftigt sich mit der Erstellung von Modellen für die horizontale Kombination von Werkzeugen, um damit am Beispiel der NLP-Aufgabe der Named Entity Recognition (NER) die Verbesserungsmöglichkeiten zu zeigen. Der TextImager bietet für jede NLP-Aufgabe mehrere Werkzeuge an, die auf der selben Datengrundlage trainiert worden sind, allerdings unterschiedliche Ergebnisse liefern können. Das heißt, dass jedes der Werkzeuge eine Teilmenge der Daten richtig verarbeitet und gleichzeitig bei einer anderen Teilmenge Fehler macht. Um eine möglichst große Teilmenge der Daten richtig zu verarbeiten, bedarf es daher einer horizontalen Kombination der Werkzeuge. Dafür wurden Machine-Learning-basierte Voting-Mechanismen, namens `LSTMVoter` und `CRFVoter`, entwickelt, die eine Kombination der Ausgaben einzelner NLP-Werkzeuge ermöglichen, so dass bessere Datenteilergebnisse erzielt werden können. In dieser Arbeit wird der Mehrwert der `Voter` am Beispiel der NER Aufgabe gezeigt, deren Ergebnisse wieder in die TextImager Toollandschaft einfließen.

Der dritte und letzte Teil dieser Arbeit beschäftigt sich mit der orthogonalen Kombination von TextImager Werkzeugen, um die Verb Sense Disambiguation (VSD) zu bewerkstelligen. Verben verbinden Handlungsträger und sind somit der organisatorische Kern der Aussagen in Texten. In dieser Arbeit wird der CL Frage nachgegangen, wie Verbsinne zu modellieren sind, um diese computational disambiguieren zu können. Verbsinne haben einen syntagmatisch-paradigmatischen Zusammenhang mit umgebenden Wörtern. Daher bedarf es einer Vorverarbeitung auf mehreren linguistischen Ebenen und somit einer orthogonalen Kombination von NLP Werkzeugen, um eine Disambiguierung der Verben auf computationaler Ebene bewerkstelligen zu können. Durch die abgebildete NLP-Landschaft von TextImager ist es nun möglich, diese Vorverarbeitungsschritte durchzuführen, um die Informationen zu induzieren, die für das VSD benötigt werden. Das neu entwickelte NLP Werkzeug für das VSD wurde in die TextImager-Toollandschaft integriert, was die Analyse einer weiteren linguistischen Ebene ermöglicht.

In der vorliegenden Arbeit wird ein Framework vorgestellt, das die NLP-Toollandschaft homogenisiert und dabei cluster-basiert arbeitet. Es werden Methoden zur Kombination der integrierten Werkzeuge implementiert, um die Analyse einer speziellen Sprachebene zu verbessern oder Werkzeuge zu entwickeln, die neue Sprachebenen erschließen.

Abstract

The World Wide Web is increasing the number of freely accessible textual data, which has led to an increasing interest in research in the field of computational linguistics (CL). This area of research addresses theoretical research to answer the question of how language and knowledge must be represented in order to understand and produce language. For this purpose, mathematical models are being developed to capture the phenomena at various levels in human languages. Another field of research experiencing an increase in interest that is closely related to CL is Natural Language Processing (NLP), which is primarily concerned with developing effective and efficient data structures and algorithms that implement the mathematical models of CL.

With increasing interest in these areas, NLP tools are rapidly and frequently being developed incorporating different CL models to handle different levels of language. The open source trend has benefited all those in the scientific community who develop and use these tools. Due to yet undefined I/O standards for NLP, however, the rapid growth leads to a heterogeneous NLP landscape in which the specializations of the tools cannot benefit from each other because of interface incompatibility. In addition, the constantly growing amount of freely accessible text data requires a high-performance processing solution. This performance can be achieved by horizontal and vertical scaling of hardware and software. For these reasons the first part of this thesis deals with the homogenization of the NLP tool landscape, which is achieved by a standardized framework called `TextImager`. It is a cloud computing based multi-service, multi-server, multi-pipeline, multi-database, multi-user, multi-representation and multi-visual framework that already provides a variety of tools for various languages to process various levels of linguistic complexity. This makes it possible to answer research questions that require the processing of a large amount of data at several linguistic

levels.

The integrated tools and the homogenized I/O data streams of the TextImager make it possible to combine the built-in tools in two dimensions: (1) the horizontal dimension to achieve NLP task-specific improvement (2) the orthogonal dimension to implement CL models that are based on multiple linguistic levels and thus rely on a combination of different NLP tools.

The second part of this thesis therefore deals with the creation of models for the horizontal combination of tools in order to show the possibilities for improvement using the example of the NLP task of Named Entity Recognition (NER). The TextImager offers several tools for each NLP task, most of which have been trained on the same training basis, but can produce different results. This means that each of the tools processes a subset of the data correctly and at the same time makes errors in another subset. In order to process as large a subset of the data as possible correctly, a horizontal combination of tools is therefore required. Machine learning-based voting mechanisms called `LSTMVoter` and `CRFVoter` were developed for this purpose, which allow a combination of the outputs of individual NLP tools so that better partial data results can be achieved. In this thesis the benefit of `Voter` is shown using the example of the NER task, whose results flow back into the TextImager tool landscape.

The third part of this thesis deals with the orthogonal combination of TextImager tools to accomplish the verb sense disambiguation (VSD). The CL question is investigated, how verb senses should be modelled in order to disambiguate them computatively. Verbsenses have a syntagmatic-paradigmatic relationship with surrounding words. Therefore, preprocessing on several linguistic levels and consequently an orthogonal combination of NLP tools is required to disambiguate verbs on a computational level. With TextImager's integrated NLP landscape, it is now possible to perform these preprocessing steps to induce the information needed for the VSD. The newly developed NLP tool for the VSD has been integrated into the TextImager tool landscape, enabling the analysis of a further linguistic level.

This thesis presents a framework that homogenizes the NLP tool landscape in a cluster-based way. Methods for combining the integrated tools are implemented to improve the analysis of a specific linguistic level or to develop tools that open up new linguistic levels.

Contents

1. Danksagung 3

2. Abstract 5

3. Introduction 13

3.1. Motivation 13

3.2. Contributions 19

4. Publications 33

4.1. TextImager: a Distributed UIMA-based System for NLP 33

4.1.1. abstract 33

4.1.2. Introduction 33

4.1.3. Related Work 34

4.1.4. System Architecture of TextImager 35

4.1.5. Future Work 38

4.1.6. Scope of the Software Demonstration 40

4.2. Evaluating and Integrating Databases in the Area of NLP 41

4.2.1. abstract 41

4.2.2. Introduction 41

4.2.3. Related Work 43

4.2.4. Database Management Systems 44

4.2.5. Experiments 47

4.2.6. System Demonstration of MDBMS 51

4.2.7. Conclusion and Future Work 52

4.3. Der TextImager als Front- und Backend für das verteilte NLP von Big Digital Humanities Data 55

4.3.1. abstract 55

4.4. TextImager as an interface to BeCalm 59
4.4.1. abstract 59
4.4.2. Introduction 59
4.4.3. Systems description and methods 60
4.4.4. Discussion 63
4.5. LSTMVoter: chemical named entity recognition using a conglomerate of sequence labeling tools 65
4.5.1. abstract 65
4.5.2. Introduction 66
4.5.3. Materials and Methods 67
4.5.4. Results 75
4.5.5. Conclusions 76
4.6. CRFVoter: Gene and Protein Related Object Recognition Using a Conglomerate of CRF-Based Tools 77
4.6.1. abstract 77
4.6.2. Introduction 78
4.6.3. Methods 80
4.6.4. System description 82
4.6.5. Results 94
4.6.6. Conclusion 97
4.7. German Verb Sense Disambiguation 99
4.7.1. Introduction 99
4.7.2. Materials and Methods 100
4.7.3. Verb Sense Disambiguation 117
4.7.4. Conclusion and Future Work 121

5. Concluding Remarks 123

A. Appendix 160

A.1. Zusammenfassung 160
A.2. Verbs 164
A.2.1. Skinners Law Evaluation 164
A.2.2. GermaNet Sense Mappings to Super Senses 168

3. Introduction

3.1. Motivation

In recent years, Big Data has evolved into a new paradigm that provides data and opportunities to improve and enable research applications in business, science and technology. At the same time, Big Data presents the challenges of storing, transporting, processing, mining and delivering data. Due to the interconnection of the world, in particular through the Internet, enormous amounts of data are produced every day. According to Desjardins (2019), per day

- 500 million tweets are sent.
- 294 billion emails are sent.
- 4 petabytes of data are created on Facebook.
- 65 billion messages are sent on WhatsApp.
- 1,5 million Wikipedia page edits containing 1,4 gigabytes data are made.

The majority of this data is unstructured, which means that no predefined data model is available for this information. Examples of unstructured data include audio, video, image files and, as initially mentioned, unstructured text from email messages, Web pages, Facebook and WhatsApp messages, or online articles. Some of them contain valuable information that can be lost in the flood of data. In order to counteract this loss, the data must be prepared in such a way that they become machine-readable. Data mining (Hand 2006; X. Wu et al. 2014), Natural Language Processing (NLP) (Aggarwal and ChengXiang Zhai 2012; Biemann and Mehler 2014), and text analysis techniques (Mehler and Köhler 2007) offer a number of methods for finding patterns, extracting meaning and creating structured data about the information. Algorithms can derive inherent structures from text data, for example by analyzing word morphology, sentence syntax, and other small and large structures. Unstructured information can then be enriched with these derived data in a structured form, for example to enable a semantic search for relevant information. It is therefore necessary to develop specific computational workflows to structure the unstructured data in text documents in order to process the millions of documents that would not be possible with manual approaches.

In the following, Wikipedia as an unstructured data source is used to illustrate the immense time and computing effort involved in structuring data. It requires a variety of operations when structuring the unstructured text of Wikipedia. These text data are processed on different linguistic levels using NLP techniques. For this purpose a part of the German Wikipedia, namely the articles of the biology category[1], is processed on ten linguistic levels listed in Table 3.2. Table 3.1 shows statistics on the processed corpus.

Table 3.1.: Number of annotations per annotation layer.

Layer	**Count**
Articles	126 467
Tokens	72 173 624
Lemma	1 901 373
Wordforms	2 207 708
Syntactic Words	3 708 560
Named Entity	3 650 992
Time	1 199 562

According to Perry (2017) the introduced corpus cannot be considered as Big Data. However, even with such a relatively small amount of data, we will show the large computing power required.

Many downstream NLP tasks, such as *Question Answering* and *Semantic Search*, require that the underlying unstructured documents are processed through a variety of pre-processing steps and enriched with additional information. Table 3.2

[1] https://de.wikipedia.org/wiki/Kategorie:Biologie

lists the most common pre-processing steps in NLP. During tokenization, texts are broken down into their document structure, where paragraphs, sentences and words are identified. These information are fundamental for all further processing steps, because, apart from individual characters, they form atomic units. Through lemmatization, words are then transformed into their basic form. For a semantic search, the recognition of word classes (part of speech) is evident, since this is the first step of disambiguating words. Furthermore, it is necessary to divide actors into classes. This is done with named entity recognition (NER) and time recognition, where multi-word expressions are grouped into predefined categories such as person, location, organization or time units. In order to be able to connect the individual actors, dependency parsing and semantic role labeling are required. Coreference resolution is necessary in order to be able to analyse the once introduced actors coherently over the entire text span. Wikification is used to clearly disambiguate the actors, whereby actors are mapped to an external knowledge resource, in this case Wikipedia, in order to induce additional external knowledge into the processed texts.

In a pilot project, state-of-the-art tools for preprocessing documents implemented by DKPro (Eckart de Castilho and Gurevych 2014) were used to analyze the corpus introduced above. Table 3.2 shows the results achieved in terms of time and storage complexity. It becomes clear that the processing of the documents takes too long even for a relatively small corpus.

In total, the sequential processing of Wikipedia articles from the biology category would take 71,75 days. In this time Wikipedia has already grown again by 5 %[2].

For this reason it is necessary that data can be processed faster. In order to be able to achieve this, a system must be developed that can operate decentralized. It must be scalable, both horizontally and vertically. It must run on dynamically many servers with dynamically many instances per tool. Tools that are not dependent on each other must be able to run in parallel to prevent bottle-

[2] `https://stats.wikimedia.org/v2/#/de.wikipedia.org/content/pages-to-date/normal|line|2019-03-01~2019-11-01|~total|monthly`

Table 3.2.: Statistics about the time and space complexity of the processed corpus. For each of the 10 NLP tasks the required time and storage is listed.

	Step	Days	GB
1	Tokenize	0,41	82,48
2	Lemmatization	1,07	97,70
3	POS Tagging	0,98	94,42
4	Named Entity Recognition	0,87	9,06
5	Dependency Parsing	22,38	225,46
6	Time Recognition	3,65	11,95
7	Sentiment Analysis	1,00	7,13
8	Semantic Role Labeling	1,21	8,09
9	Wikification	15,99	26,40
10	Coreference	24,19	6,74
	$\sum$	71,75	569,43

necks.

Cloud Computing provides basic support in addressing the challenges posed by shared computing resources such as computing, storage, networking and analytic software; the application of these resources has led to impressive advances in Big Data processing. For this reason, a Cloud Computing framework was developed in the course of this work, which is based on the following principles:
(1) Multi-Service-System, (2) Multi-Server-System, (3) Self-Orchestration-System, (4) Multi-Database-System, (5) Authority-Management-System, (6) Multi-Representation-System. Section 3.2 describes the importance and how these principles have been implemented.

One core task of NLP is missing in the pipeline presented, namely the interpretation of the relationship between the individual actors or participants in a text span. In order to answer this question, the organizational core of a statement must be identified and semantically interpreted. Verbs name events or states with participants and thus make them the organizational core of the sentence, so their meaning is the key to sentence meaning. Therefore, disambiguation of verbs plays a key role in automatic text comprehension, and is therefore a core task of

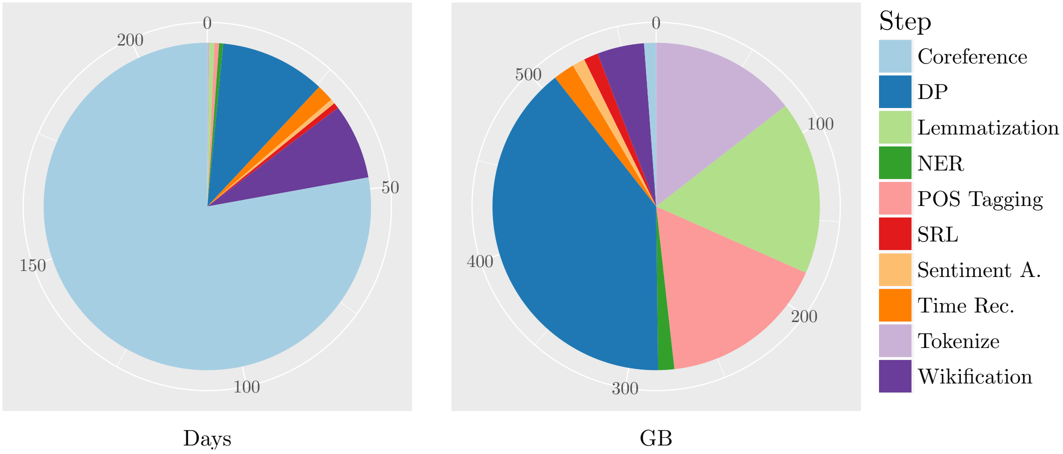


Figure 3.1.: The time and space analysis of the Wikipedia articles in the Biology category after processing by the ten NLP steps can be seen. The processing takes more than 71 days and takes more than 500 GB. In this time Wikipedia has already grown by 30 %.

NLP.

The task of verb sense disambiguation (VSD) is to automatically assign a predefined sense to a polisemic verb based on its context. The goal of an automatic VSD system is to tag unseen examples with a sense derived from a sense inventory or dictionary.

Understanding the semantics of words in natural language text is an important task for automatic knowledge acquisition. For many semantic downstream tasks, such as machine translation (Vickrey et al. 2005; Sudarikov et al. 2016; Neale et al. 2016), semantic parsing (Shi and Rada Mihalcea 2005; Giuglea and Moschitti 2006), information retrieval (Zhong and H. T. Ng 2012; Chifu et al. 2015; H. T. Ng 2011) and question answering (Hung et al. 2005), it is therefore indispensable to carry out such a disambiguation.

Due to the relevance of this topic, several shared tasks (Edmonds and Cotton 2001; Snyder and M. Palmer 2004; Pradhan et al. 2007; Navigli, Jurgens, and Vannella 2013) have already been organized. However, the focus was mainly on the English language. Henrich (2015) carried out the most extensive work on VSD for German to date. Various manually annotated and semi-automatically annotated corpora were developed, on the basis of which machine learning methods were trained. For this, a sense inventory was developed, which connects entries hierarchically at sense level with semantic relations. Two machine learning strategies were used, supervised and unsupervised. The best-performing supervised method of Henrich (2015) achieved an F-Score of 80 %, but only on 79 verbs. The unsupervised variant works on the whole sense inventory, but with an F-score of 55 % which is on average almost a random distribution with 2,3 senses per lemma.

Therefore, a part of this dissertation is to make a substantial contribution to VSD for German. The task of the VSD was divided into two parts. On the one hand, a corpus was developed that covers a large number of verbs, whereby supervised methods can be sufficiently well trained. For this, semi-automatic methods were developed to achieve a faster and cheaper annotation of the data, namely by adapting the Skinner's Law, by translation and utilizing language models. On the other hand, methods have been developed that use the pre-processing steps of the TextImager to automatically make sense distinctions of verbs at a syntactic level. These developments led to a state-of-the-art VSD system for German, covering 80 % of verb tokens. The materials, methodologies and results of this VSD system are presented in Section 4.7.

3.2. Contributions

The computational performance requirements have increased dramatically in recent years. This also applies to many language processing tasks, as the ever-growing amount of textual information has to be processed within a reasonable time frame. This scenario has led to a paradigm shift in computer architectures and large-scale data processing strategies in Natural Language Processing. Another challenge for all existing infrastructures is making new methods available for more and more NLP tasks. As of today there are several thousand freely available NLP projects[3]. This number is growing steadily. Consequently, there is a need to develop flexible architectures that can cope with this growth. The question is along which dimensions this flexibility is to be guaranteed.

To answer this question, TextImager is introduced as an example of systems of algorithmic (automatic) analysis and (human) interpretation of the analysis results (Mehler, Hemati, Gleim, et al. 2018). The TextImager, which describes the above requirements in six criteria, is presented below:

Multi-Service-System

Developments in text technology and the digital humanities are in a state of rapid change and are reflected in the variety of applications, platforms and frameworks being developed. In recent years there has been a strong shift towards web services and web applications (Sanders, Jr., and MacDonald 2008; Maurizio et al. 2008; Conlon et al. 2008; Mehler, Gleim, Waltinger, et al. 2009; M. Hinrichs, Zastrow, and E. W. Hinrichs 2010; Mehler, Schwandt, et al. 2011; Gleim, Mehler, and Ernst 2012; Sahami, Eckart, and Heyer 2019). Some of the main advantages are the simple or omitted software installation as well as the platform independence. But the real benefits are the opportunities to share and use resources and services online (Richardson and Ruby 2007; Gleim, Mehler, and Ernst 2012; Erl 2016). The concept of Service Oriented Architectures (SOAs) has been proposed as a means of integrating isolated services to provide more complex functionality.

[3] https://github.com/topics/nlp

SOAs are not standardized, but must be designed according to the requirements of the respective application area.

It is also conceivable to merge existing systems and development environments to exchange work packages with each other in order to exploit specialisation advantages. In order for the specializations of the different systems to benefit from each other, they have to be adapted to allow data streams to be exchanged between the systems. For this purpose, a standardized input/output format must be created which allows systems from different provinces to became compatible in SOAs.

For this purpose, TextImager introduces and implements a type system[4], which forms the interface between all the integrated components. A type system defines the different types of objects that can be discovered in documents by the TextImager, such as tokens, lemmas, POS and other linguistic objects on other linguistic levels.

TextImager's integrated components exchange their data via the UIMA Common Analysis System (CAS), which provides access to the type system by forming an object schema (see Figure 3.2) (Ferrucci and Lally 2004). The CAS object is also the container that manages Subjects of Analysis (Sofas), which represents an unstructured artifact, i.e. a document, that is processed in the TextImager pipeline. Each tool in the system must define its input and output types. This definition is then used as the basis for orchestrating the tools in concurrent pipelines, in line with the general idea of an SOA that describes how services should be orchestrated, rather than describing the services themselves.

The process for registering a new tool in TextImager has been optimized so that only one interface, namely the AnalysisComponent, needs to be implemented and the input and output types for this tool need to be specified. Analysis components are the primitive building blocks. This is the common interface for all components that take a CAS as input and may produce CASes as output.

[4] `https://github.com/texttechnologylab/textimager-uima/tree/master/textimager-uima-types`

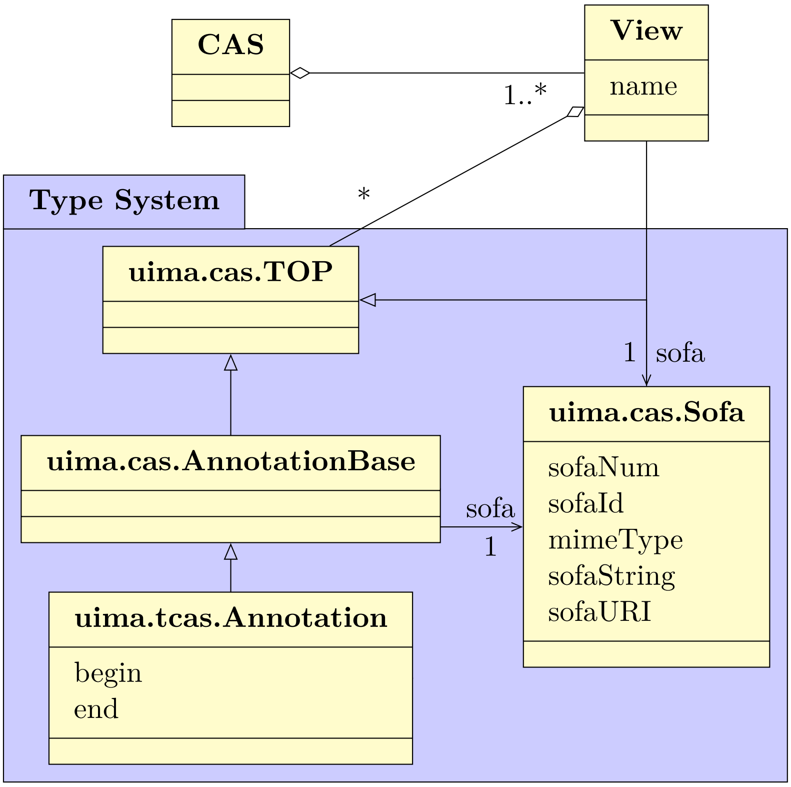


Figure 3.2.: Architecture of Common Analysis System (CAS) accessing the type system. A type system defines the different types of objects that can be discovered in documents and stored in CASes.

The goal of TextImager is to integrate the latest state-of-the-art tools per language into the NLP landscape. Table 3.3 shows the current tool list for the two dimensions *language* and *NLP Task*.

TextImager offers all of its tools as web services, which can be used independently of platforms and programming languages. The standardized REST interface[5] or the Java implementation[6] can be used for this.

[5] https://textimager.hucompute.org/rest/doku/

[6] https://github.com/texttechnologylab/textimager-client

Table 3.3.: Number of NLP service per language available at TextImager.

	en	de	es	fr	la	nl	pt	zh	it	da	ar	Other	$\sum$
Tokenize	6	4	4	4	3	2	2	1	2	3	3	10	44
Lemmatization	10	4	4	2	5	1	4	0	2	0	0	11	43
POS Tagging	19	11	5	4	5	4	4	4	2	2	2	13	75
NER	15	7	4	4	0	4	1	0	1	1	0	14	51
Parsing	7	3	3	4	0	0	0	5	2	2	3	9	38
Time Rec.	1	1	1	1	0	1	1	0	1	1	1	1	10
Sentiment	3	1	0	0	0	0	0	0	0	0	0	3	7
SRL	3	2	0	0	0	0	0	0	0	0	0	2	7
Wikification	3	3	1	0	0	0	0	0	0	0	0	2	9
Coreference	4	3	0	0	0	0	0	0	0	0	0	2	9
$\sum$	71	39	22	19	13	12	12	10	10	9	9	67	293

Multi-Server-System

More and more computing capacity is needed to coordinate the work of NLP tools on ever larger amounts of data. TextImager meets this challenge by distributing its services across a basically unlimited number of servers in such a way that an n:m relation of services and servers with shared computing resources is created. The Java Messaging Service (JMS) (Hapner et al. 2002) implemented by Apache UIMA-AS is used for this purpose, that enables communication between different components of a distributed application that can be located on different servers. JMS implements a point-to-point communication system. This communication type is based on the concept of message queues, senders and receivers. Each service in the distributed server system is assigned an input queue and an output queue. To create multiple instances of a component the instances connect to the same service input queue. The instances receive work units from this queue, CASes to be more precise. After processing, the result is returned to an output queue (Hemati, Mehler, Uslu, and Abrami 2019). This approach makes it possible to switch new servers and services on and off at run-time to prevent overhead.

The point-to-point architecture enables decentralization of services. Infrastructures of this kind can therefore be expanded as interfaces to multi-server systems, which are constantly growing due to decentralized initiatives and further developments. This makes it possible for users to host their own TextImager-Servers[7] and have their own services, communicate with services in order to manage the distributed execution of their annotation tasks using this infrastructure[8]. Hosting your own TextImager-Server has additionally the advantage that resources (corpora, tools, etc.) do not have to be publicly released, but still benefit from other existing resources..

Self-Orchestration

As a result of the variety of methods, more and more algorithms (e.g. conditional random fields, long short-term memory, transformer networks) and tools (e.g. MarMoT) are available for the same task (e.g. tagging), which in turn span parameter spaces. In addition, in the course of the development of computational linguistics, a rudimentary process model has established itself as a scientific discipline, which defines practiced sequence regularities of such tasks (Mehler, Hemati, Gleim, et al. 2018). Semantic Role Labeling, for example, is often based on parsing, which in turn results of lemmatization and POS tagging.

The homogenized TextImager type system enables a high degree of abstraction in the construction of such a process pipeline, since the input and output types of the respective tools define the pre-processing requirements. This makes it possible that the individual tools for processing of respective tasks can be exchanged with each other in a modular principle, whereby only the input requirements of the respective tools in the process chain have to be created beforehand. Tasks that are independent of each other can run concurrently. The process or workflow model for such a pipeline structure must be capable of (1) representing tasks that generate output types; (2) representing tools that request input types and generate output types; (3) orchestrating tools into a flow so that parallel subtasks can be executed concurrently.

Petri nets (Reisig 1985) can be used to implement a workflow model that covers

[7] `https://github.com/texttechnologylab/textimager-server`

[8] `https://github.com/texttechnologylab/textimager-client`

all three requirements. More precisely, a Place/Transition Petri net (Desel and Reisig 1996) is implemented, which is defined as follows:
$PN = (P, T, F, W, M_0)$
where:

- $P = p_1, p_2, ..., p_m$ is a finite set of places, in our case types from the type system;
- $T = t_1, t_2, ..., t_m$ is a finite set of transitions, in our case NLP tools;
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (pre- and post-conditions);
- $W : F- > 1, 2, 3, ...$ is a weight function
- $M_0 : P- > 0, 1, 2, 3, ...$ is the initial marking

Graphically, Petri nets are represented as bipartite graphs, with two disjoint types of nodes: places (P) by circles and transitions (T) by rectangles linked by arcs. In the case of the TextImager process model, places define types in the type system (Token, Lemma, POS, etc) and transitions represent NLP tools (MarMotLemmatizer, MarMotPos, etc). The tools have different types as pre-condition and post-condition, which means that they request input types and produce output types. The task of the process model is now to execute the selected tools in such a way that as little overhead as possible is produced by processing as many tools as possible concurrently. The elementary building blocks of Petri Nets from Figure 3.3, namely (a) sequence, (b) fork, (c) synchronization, (d) choice, and (e) merging, are used to realize the complex pipeline structure of TextImager.

Figure 3.4 shows the entire NLP landscape of TextImager represented by a Petri net. Figure 3.5 shows the subgraph for creating a pipeline, that results in *language, paragraph, sentence, token, pos, entity, lemma* and *dependency* types for a given German document. Tools (represented by transitions (rectangles)) that are at the same depth of the tree can run concurrently as they have no dependent types.

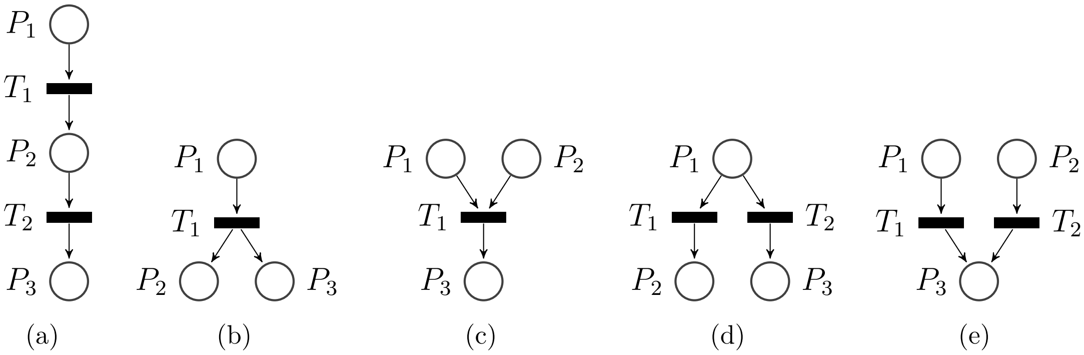


Figure 3.3.: Elementery Petri-Net Structures

Multi-Database-System

With the growth of tools, the diversity of representation possibilities of the processed data increases, whether in the form of document-oriented (XML-based) tree structures, data-oriented graph structures or numerical distributions. This diversified data can be stored on distributed files in the appropriate format (XML, TEI, TSV, JSON, etc.). However, this has the disadvantage that the data cannot be searched efficiently. Databases offer a way out, since the stored data can be indexed efficiently and made searchable. However, there is no database solution that enables all of the forms of data representation mentioned above. To overcome this gap, a multi-DB system is required that can manage multiple database management systems (DBMS) that are optimized for the characteristics of each of these representations. Hemati, Mehler, Uslu, Baumartz, et al. (2018) shows, that for example, graph databases such as Neo4j are suitable for the representation of simple directed graphs, while MongoDB is better equipped for the management of document-oriented structures. Blazegraph, on the other hand, is suitable for the management of RDF-based data. TextImager also addresses this aspect of dynamization. The model for this is the openness that already characterizes multi-server, multi-service systems: The aim is to continuously integrate new databases into the system in order to be able to represent new representation possibilities.

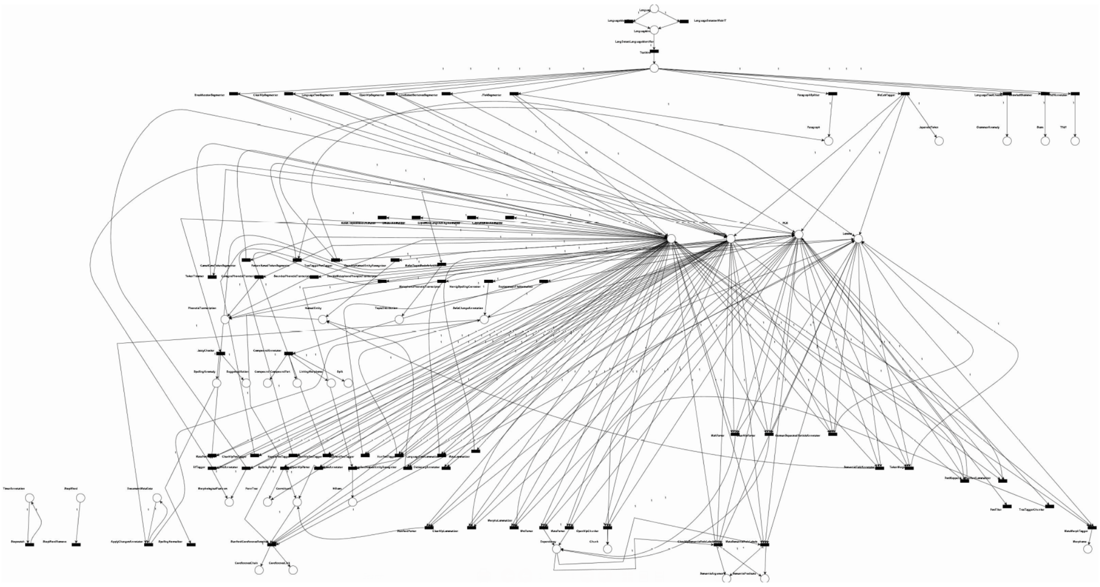


Figure 3.4.: Illustration of the entire Textimager tool landscape and the interdependence of the tools.

Authority Management

Distributed server systems that perform reproducible data-oriented work require the ability to offer different rights to the analysis and annotation of the underlying data. For this, TextImager uses the rights management system of the eHumanities desktop (Gleim, Mehler, and Ernst 2012). This enables different users to have different access rights to resources, services and servers.

Multi-Representation-System

By integrating the different frameworks, many formats have to be read in and produced. Diverse input and output file formats ranging from simple text, over various corpus formats (PDF, HTML, CoNLL, TIGER-XML, BNC-XML, TCF, etc.), to tool specific formats and database formats (Neo4j, Cassandra, SQL, MongoDB, etc.). In addition, entire Wikiditions (Mehler, B. Wagner, and Gleim 2016; Mehler, Gleim, Brück, et al. 2016; Hunziker et al. 2019) can be created, whereby a processed corpus is exported into the MediaWiki format and enriched

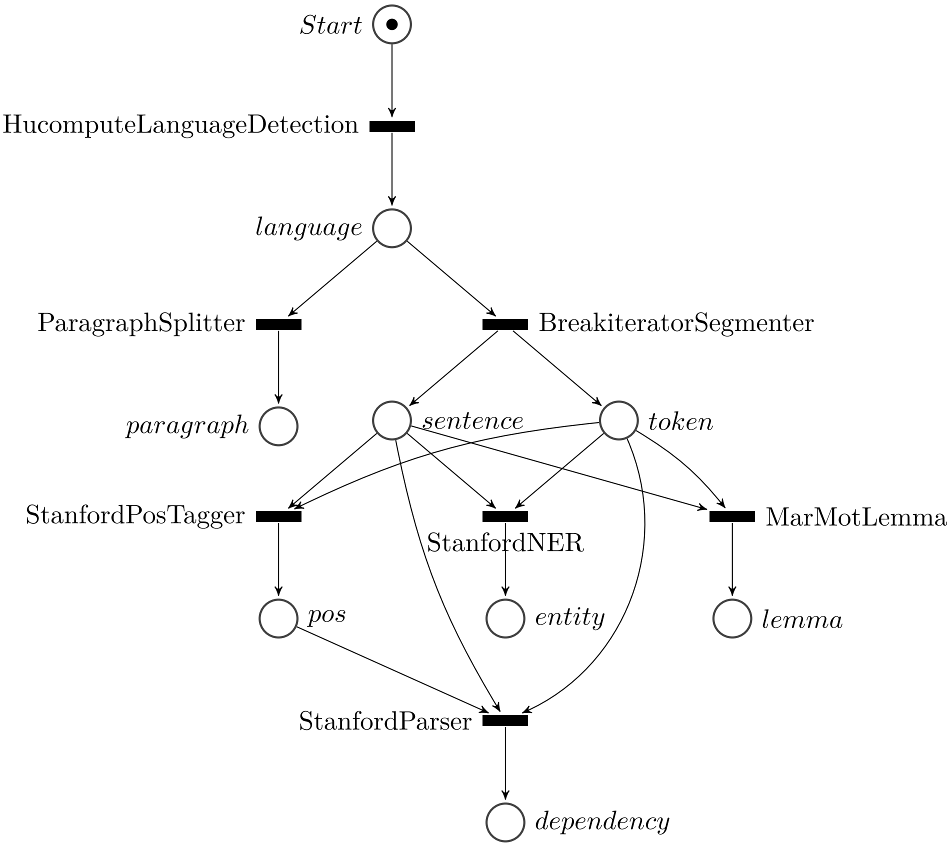


Figure 3.5.: Section of tool landscape represented by the Petri net which creates *language, paragraph, sentence, token, pos, entity, lemma* and *dependency* types. Tools (represented by transitions (rectangles)) that are at the same depth of the tree can run concurrently as they have no dependent types.

with information on different linguistic levels. Various methods for data visualization have been developed (Hemati, Uslu, and Mehler 2016; Uslu and Mehler 2018; Uslu, Mehler, and Meyer 2018). The various visualizations have been integrated into a user-friendly graphical interface, which now makes it possible to use the TextImager NLP tools without programming knowledge. This makes it possible to use NLP interdisciplinary, e.g. in the humanities scholar. This type of processing opens the way to new research questions in the humanities and in particular to various methods of answering them.

Remarks

TextImager provides NLP pipelines for the automatic analysis of textual units. Comparable to Voyant Tools (Sinclair and Rockwell 2012), it deals with the interactive visualization of textual structures in such a way that interactions with texts lead to adaptations of the visualizations entangled with them. TextImager therefore allows interaction with visualizations in order to control browsing in the input texts, as well as vice versa browsing in texts in order to achieve consistent visualizations to create textsensitive visualizations. Comparable to SpaCy (Honnibal and Montani 2017), AllenNLP (Gardner et al. 2018) and CoreNLP (Manning et al. 2014), it is a pipeline oriented framework for NLP. Similar to DKPro (Eckart de Castilho and Gurevych 2014), TextImager uses Apache UIMA to orchestrate NLP tools into pipelines. UIMA provides interfaces for the development of modular NLP components. Comparable to GATE (Cunningham et al. 2011), RapidMiner (Hofmann and Klinkenberg 2013) and WebLicht (M. Hinrichs, Zastrow, and E. W. Hinrichs 2010), the goal is to make text mining tools available on as broad a basis as possible. In principle, any freely available tool for any natural language that can be standardized in accordance with UIMA should be includable into TextImager. At the current state, TextImager has already subsumed a number of the above mentioned frameworks, including DKPro, SpaCy, AllenNLP and CoreNLP as preprocessing modules, making it possible to exchange work packages with each other in order to exploit specialisation advantages.

Table 3.4.: Comparison of TextImager to other frameworks over the 6 dimensions presented.

	Voyant	**DKPro**	**SpaCy**	**AllenNLP**	**CoreNLP**	**TextImager**
Multi-Service						x
Multi-Server						x
Self-Orchestration				x	x	x
Multi-Database						x
Authority Management	x					x
Multi-Representation	x	x	x		x	x
Open-Source		x	x	x	x	x

Table 3.4 shows the frameworks presented above according to the 6 dimensions presented so far. Each of the tools implements a subset of the dimensions, with TextImager implementing all.

Due to the various integrated tools and the homogenized I/O data streams in the form of a standardized type system it is possible to combine the built-in modules on two different dimensions (see Figure 3.6); (1) the horizontal dimension in order to achieve a task specific improvement (2) the orthogonal dimension, in order to create high level NLP tools:

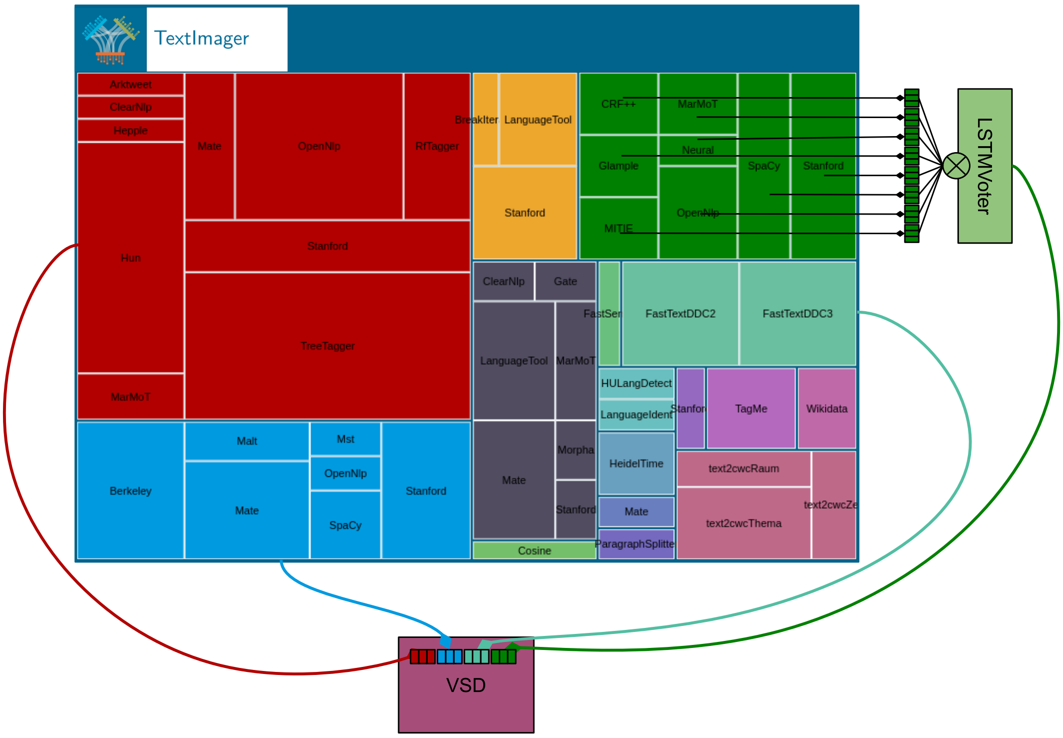


Figure 3.6.: Illustration of TextImager's module combination in horizontal (LSTMVoter) and orthogonal (VSD) dimension. The rectangles represent the integrated modules, where the size reflects the number of supported languages. Modules are color coded into task specific groups (POS, NER, parsing, etc.). The green block represents NERs combined using LSTMVoter. POS (red), NER (green), Dependency (blue), Topic labeling (turquoise) features are used and combined for the VSD system.

(1) The horizontal combination merges the outputs of modules trained and specialized for the same task. As an example, the combinatorial improvement of the NER task is carried out in this dissertation. TextImager provides many tools (StanfordPosTagger, MarMotTagger, etc.) for solving the same task (PoS-Tagging, NER, ...). The output of the respective tools can be different, whereby each tool correctly classifies a different subset of the data. A combination of these tools to a conglomerate can therefore lead to an improvement of the result. For this purpose `CRFVoter` (see Section 4.6.1) and `LSTMVoter` (see Section 4.5.1) were developed, whereby the output of the respective tools is weighted in the context of the task. As an example for the functionality, different sequence classifiers for the recognition of biochemical terms were trained and combined with the help of the two voters (Hemati and Mehler 2019a; Hemati and Mehler 2019b).

In the first step, each sequence classifier $c_m, m = 1..l$, is optimized independently on a subset of the training set, where the ith sequence t_i of length n of the set of training examples is of the form

$$t_i = \langle(\vec{x}_1, y_1), ..., (\vec{x}_n, y_n)\rangle \tag{3.1}$$

$\vec{x}_j, j = 1..n$, is a feature vector corresponding to an element in the input sequence at position j. y_j is the corresponding discrete label of the element at position j. The goal of a sequence classifier c is to approximate the function $f(j) = y_j$ where y_j is the true label to be assigned to the input stream at position j.

In the second step, the `Voters` combines each sequence classifier c_m into an ensemble classifier $c = \texttt{Voter}(\{c_1, c_2, ..., c_l\})$. The sequence of training examples is of the form

$$t_i = \langle((f_{c_1}(\vec{x}_1), f_{c_2}(\vec{x}_1), ..., f_{c_l}(\vec{x}_1)), y_1), ..., ((f_{c_1}(\vec{x}_n), f_{c_2}(\vec{x}_n), ..., f_{c_l}(x_n)), y_n\rangle \tag{3.2}$$

where $f_{c_m}(\vec{x}_j), m = 1..l, j = 1..n$, is the output label of classifier c_m computed for the input vector $\vec{x}_j$ at the jth position of the input sequence.

In this way, we train `CRFVoter` and `LSTMVoter` based on a sequence of the latter feature sets computed by pre-trained sequence labeling systems. We have shown that `CRFVoter` (Hemati and Mehler 2019a) and `LSTMVoter` (Hemati and Mehler 2019b) have by far outperformed the independent best performers on the task of biomedical named entity recognition.

(2) The orthogonal combination of the modules combines the outputs of modules solving NLP subtasks that provide orthogonal outputs at different linguistic levels in order to build downstream NLP modules. One of these downstream modules is verb sense disambiguation (VSD). A verb sense depends on various features, including morphology, syntax, and thematic classification of the verb to be disambiguated and the surrounding words. All these features were produced by the TextImager to finally build a VSD system on the combination of the outputs.

A result of this work is a state-of-the-art verb sense disambiguation system for German, alongside with the largest sense annotated corpus for German verbs (see Section 4.7). Since annotation is very time-consuming and expensive, automatic and semi-automatic methods for corpus expansion have been developed and integrated into TextImager. These include expansion through Skinner's Law (Mehler 2005), translation and language models. Many classification tasks suffer from insufficient training data. For these classification tasks, especially for semantic classification tasks, it may be useful to use our proposed expansion methods. It has been shown that such a resource is needed to disambiguate a sufficiently large number of verbs using supervised machine learning (ML). The subset of verb lemmas that covers 80% of verb tokens was annotated. The reliability and validity the data is evaluated using Randolph's Kappa. In the process, it turned out that some verbs have low inter-annotator agreement (IAA). The reason for this is that the distinction of senses in GermaNet, the used sense inventory, is too fine-grained for these verbs. In order to solve this problem, a method was developed to locate and merge these fine-grained senses into supersenses.

Various neural network-based taggers were trained and evaluated using features produced by TextImager in combination with the hand annotated training corpus. Compared to the previous state-of-the-art, the best performer of this work achieved a performance increase of 6% F-Score (80% to 86%) and is therefore state-of-the-art.

4. Publications

4.1. TextImager: a Distributed UIMA-based System for NLP

Hemati, Wahed, Tolga Uslu, and Alexander Mehler (2016). "TextImager: a Distributed UIMA-based System for NLP". In: *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference System Demonstrations, December 11-16, 2016, Osaka, Japan*, pp. 59–63. URL: `https://www.aclweb.org/anthology/C16-2013/`.

4.1.1. abstract

More and more disciplines require NLP tools for performing automatic text analyses on various levels of linguistic resolution. However, the usage of established NLP frameworks is often hampered for several reasons: in most cases, they require basic to sophisticated programming skills, interfere with interoperability due to using non-standard I/O-formats and often lack tools for visualizing computational results. This makes it difficult especially for humanities scholars to use such frameworks. In order to cope with these challenges, we present TextImager, a UIMA-based framework that offers a range of NLP and visualization tools by means of a user-friendly GUI. Using TextImager requires no programming skills.

4.1.2. Introduction

Computational humanities and related disciplines require a wide range of NLP tools to perform automatic text analyses on various levels of textual resolution.

This includes, for example, humanities scholars dealing with repositories of historical documents, forensic linguists analyzing unstructured texts of online social media to create digital fingerprints of suspects or even doctors using clinical NLP to support differential diagnosis based on physician-patient talks. However, established NLP frameworks still require basic to sophisticated programming skills for performing such analyses. This hampers their usage for users who are not sufficiently trained neither in computational linguistics nor in computer science. Further, these frameworks often lack interoperability due to using non-standard I/O-formats. We present TextImager to cope with these challenges. The longer-term goal of TextImager is to provide a platform into which any open source/access NLP tool can be integrated. To this end, TextImager provides a web-based GUI whose usage does not require any programming skills while making accessible a range of tools for visualizing results of text analyses. In order to ensure standardization and interoperability, TextImager is based on the *Unstructured Information Management Applications* (UIMA) framework. Currently, the scope of TextImager ranges from tokenizing, lemmatizing, POS-tagging, text similarity measurements to sentiment analysis, text classification, topic modeling and many more.

4.1.3. Related Work

Frameworks of computational texts analysis have already been introduced and are now common in industrial use. This includes, for example, UIMA (Ferrucci and Lally 2004), DKPro (Eckart de Castilho and Gurevych 2014), OpenNLP (OpenNLP 2010) and Gate (Cunningham et al. 2011). Note that these frameworks do not provide visualization interfaces and require versatile programming skills for set up. Thus, they cannot be recommended for being used by computationally less trained users. We provide the TextImager to cope with this problem while integrating most of the components of these frameworks. On the other hand, Voyant Tools (Bird, Klein, and Loper 2009; Ruecker, Radzikowska, and Sinclair 2011), WebNLP (Burghardt et al. 2014) and conTEXT (Khalili, Auer, and Ngomo 2014) are web-based NLP tools including visualization components.

In order to combine the best of both worlds, TextImager additionally subsumes the functionalities of these tools. It also shares functionalities with WebLicht (E. W. Hinrichs, M. Hinrichs, and Zastrow 2010). However, unlike WebLicht, TextImager is based on open UIMA and, thus, complies to an industrial standard of modeling text processing chains.

4.1.4. System Architecture of TextImager

TextImager consists of two parts, front-end and back-end. The front-end is a web application that makes all functionalities and NLP processes available in a user-friendly way. It allows users for analyzing and visualizing unstructured texts and text corpora. The back-end is a highly modular, expandable, scalable and flexible architecture with parallel processing capabilities.

Back-end

Figure 4.1 shows the architecture of TextImager. Every NLP component of TextImager implements a UIMA interface. Every UIMA compatible NLP-component can easily be integrated into TextImager. Even modules not compatible with UIMA can be integrated with just a slight effort. Amongst others, we have integrated DKPro (see Section 4.1.3), which offers a variety of UIMA-components. We also integrated the *UIMA Asynchronous Scaleout* (UIMA-AS)[1] add-on. TextImager allows users for dynamically choosing NLP components in a pipeline. To this end, we extended UIMA-AS by initiating compo-

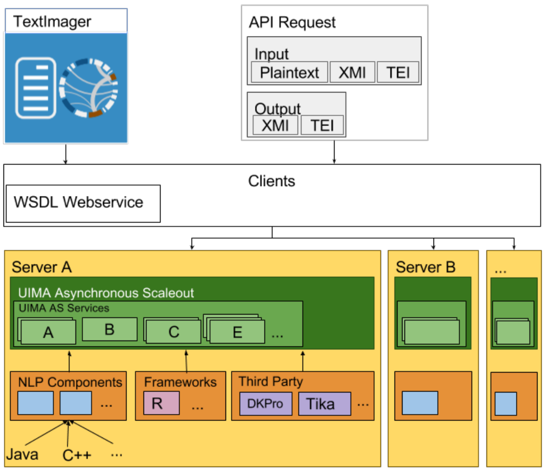


Figure 4.1.: TextImager's back-end.

[1] https://uima.apache.org/doc-uimaas-what.html

nents without XML descriptors by means of uimaFIT[2]. We extended this framework by allowing for dynamic instantiations of pipelines. These extensions make our framework highly flexible, adaptive and extensible during runtime.

All TextImager components are configured as UIMA-AS services, which may run standalone or in a pipeline. All services are located on servers to allow for communication among them. Note that we are not limited to run these components on a single server; rather, they can be distributed among different servers (see Figure 4.1). We developed a mechanism that automatically selects and acquires components and their resources: it arranges components into pipelines and grants the ability to parallelize them. Thus, components that do not depend on each other can run in parallel. For this we developed an advanced UIMA flow controller. Take the examples displayed in Fig.4.2: suppose that vertices in these examples denote NLP components; suppose further that the corresponding arcs denote interdependencies between these components. In Fig. 4.2a, the components C_1, C_2 and C_3 do not depend on each other. Thus, they can run in parallel. In Fig. 4.2b, components C_1 and D_1 do not depend on each other, but on C and D, respectively. Thus, C and D can run in parallel as can do the components C_1 and D_1. In Fig. 4.2c, C depends on C_1, C_2 and C_3. Thus, running C has to wait on the termination of C_1, C_2 and C_3. Within TextImager, dependency hierarchies of components as exemplified by these three examples are generated from information provided by each of the components supposed that their input and output types have been defined appropriately (cf. the class specifications of type `org.apache.uima.fit.descriptor.TypeCapability`). In this way, TextImager allows for realizing a wide range of processing chains.

One advantage of our framework is that it does not rely on a central repository. Rather, TextImager can be distributed across multiple servers. This allows developers for setting up their own TextImager server and to distribute their own NLP tools within the TextImager ecosystem.

TextImager can be used within a web application that offers a graphical user interface. Alternatively, TextImager can be used via a WSDL webservice client.

[2] `https://uima.apache.org/uimafit.html`

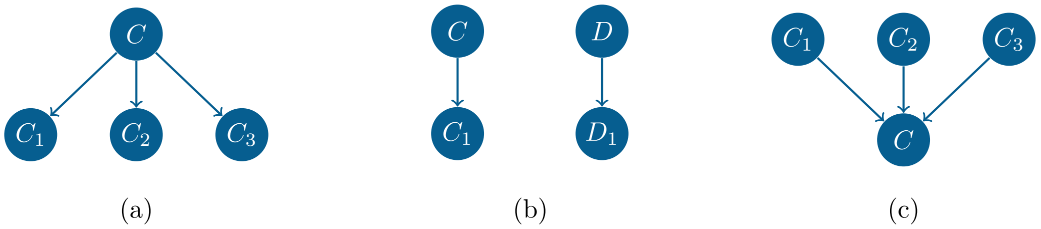


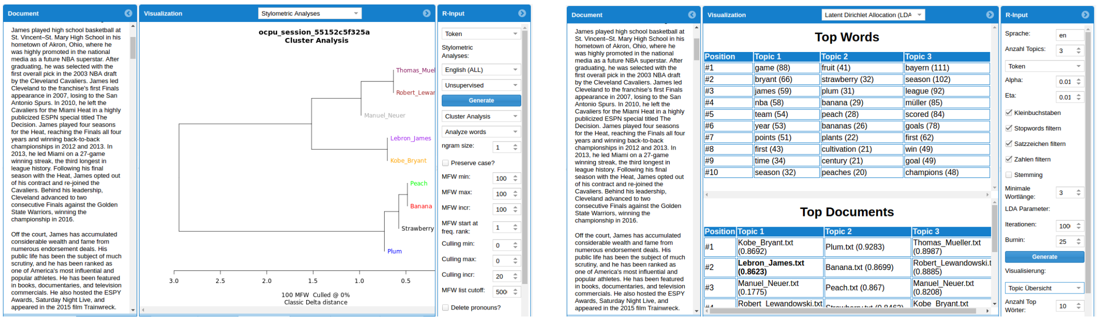
Figure 4.2.: Component dependency types.

Front-end

The front-end gives access to all NLP tools integrated into TextImager without requiring any programming skills. This is done by means of a GUI that even provides three-dimensional text visualizations (see Figure 4.4b). All visualizations are interactive in the sense of allowing for focusing and contextualizing results of text analysis (e.g., the macro *reference distribution of sentence similarity across multiple documents* exemplified in Figure 4.4d). The GUI contains a text and a visualization panel. One of TextImager's guiding principles is to enable *bidirectional interactivity*. That is, any interaction with the visualization panel is synchronized by automatically adjusting the content of the text panel and vice versa. The front-end is based on *Ext JS*, a JavaScript framework for building interactive cross platform web applications. The visualizations are done by means of *D3.js*[3] and *vis.js*[4] to enable browser-based visualizations while handling large amounts of data.

Figure 4.4 exemplifies TextImager. With a focus on close reading, TextImager supports the interpretation of single texts by determining, for example, their central topics or by depicting their unfolding from constituent to constituent (see Figure 4.4g, 4.4a, 4.4h). Regarding distant reading (Jänicke et al. 2015), TextImager provides more abstract overviews of the content of text corpora. Here, visualizations provide summary information as exemplified in Figure 4.4b, 4.4c, 4.4d, 4.4f.

[3] `https://www.d3js.org`
[4] `http://visjs.org`



Document

James played high school basketball at St. Vincent–St. Mary High School in his hometown of Akron, Ohio, where he was highly promoted in the national media as a future NBA superstar. After graduating, he was selected with the first overall pick in the 2003 NBA draft by the Cleveland Cavaliers. James led Cleveland to the franchise's first Finals appearance in 2007, losing to the San Antonio Spurs. In 2010, he left the Cavaliers for the Miami Heat in a highly publicized ESPN special titled The Decision. James played four seasons for the Heat, reaching the Finals all four years and winning back-to-back championships in 2012 and 2013. In 2013, he led Miami on a 27-game winning streak, the third longest in league history. Following his final season with the Heat, James opted out of his contract and re-joined the Cavaliers. Behind his leadership, Cleveland advanced to two consecutive Finals against the Golden State Warriors, winning the championship in 2016.

Off the court, James has accumulated considerable wealth and fame from numerous endorsement deals. His public life has been the subject of much scrutiny, and he has been ranked as one of America's most influential and popular athletes. He has been featured in books, documentaries, and television commercials. He also hosted the ESPY Awards, Saturday Night Live, and appeared in the 2015 film Trainwreck.

Visualization: Stylometric Analyses

ocpu_session_55152c5f325a
Cluster Analysis

100 MFW Culled @ 0%
Classic Delta distance

R-Input

Token

Stylometric Analyses: English (ALL)

Unsupervised

Generate

Cluster Analysis

Analyze words

ngram size: 1

□ Preserve case?

MFW min: 100

MFW max: 100

MFW incr: 100

MFW start at freq. rank: 1

Culling min: 0

Culling max: 0

Culling incr: 20

MFW list cutoff: 500

□ Delete pronouns?

(a) Stylometric R-Module

Document

James played high school basketball at St. Vincent–St. Mary High School in his hometown of Akron, Ohio, where he was highly promoted in the national media as a future NBA superstar. After graduating, he was selected with the first overall pick in the 2003 NBA draft by the Cleveland Cavaliers. James led Cleveland to the franchise's first Finals appearance in 2007, losing to the San Antonio Spurs. In 2010, he left the Cavaliers for the Miami Heat in a highly publicized ESPN special titled The Decision. James played four seasons for the Heat, reaching the Finals all four years and winning back-to-back championships in 2012 and 2013. In 2013, he led Miami on a 27-game winning streak, the third longest in league history. Following his final season with the Heat, James opted out of his contract and re-joined the Cavaliers. Behind his leadership, Cleveland advanced to two consecutive Finals against the Golden State Warriors, winning the championship in 2016.

Off the court, James has accumulated considerable wealth and fame from numerous endorsement deals. His public life has been the subject of much scrutiny, and he has been ranked as one of America's most influential and popular athletes. He has been featured in books, documentaries, and television commercials. He also hosted the ESPY Awards, Saturday Night Live, and appeared in the 2015 film Trainwreck.

Visualization: Latent Dirichlet Allocation (LDA

Top Words

Position	Topic 1	Topic 2	Topic 3
#1	game (88)	fruit (41)	bayern (111)
#2	bryant (66)	strawberry (32)	season (102)
#3	james (59)	plum (31)	league (92)
#4	nba (58)	banana (29)	müller (85)
#5	team (54)	peach (28)	scored (84)
#6	year (53)	bananas (26)	goals (78)
#7	points (51)	plants (22)	first (62)
#8	first (43)	cultivation (21)	win (49)
#9	time (34)	century (21)	goal (49)
#10	season (32)	peaches (20)	champions (48)

Top Documents

Position	Topic 1	Topic 2	Topic 3
#1	Kobe_Bryant.txt (0.8692)	Plum.txt (0.9283)	Thomas_Mueller.txt (0.8987)
#2	**Lebron_James.txt (0.8623)**	Banana.txt (0.8699)	Robert_Lewandowski.t (0.8885)
#3	Manuel_Neuer.txt (0.1775)	Peach.txt (0.867)	Manuel_Neuer.txt (0.8208)
#4	Robert_Lewandowski.txt	Strawberry.txt (0.8462)	Kobe_Bryant.txt

R-Input

Sprache: en

Anzahl Topics: 3

Token

Alpha: 0.01

Eta: 0.01

☑ Kleinbuchstaben

☑ Stopwords filtern

☑ Satzzeichen filtern

☑ Zahlen filtern

□ Stemming

Minimale Wortlänge: 3

LDA Parameter:

Iterationen: 100

Burnin: 25

Generate

Visualisierung:

Topic Übersicht

Anzahl Top Wörter: 10

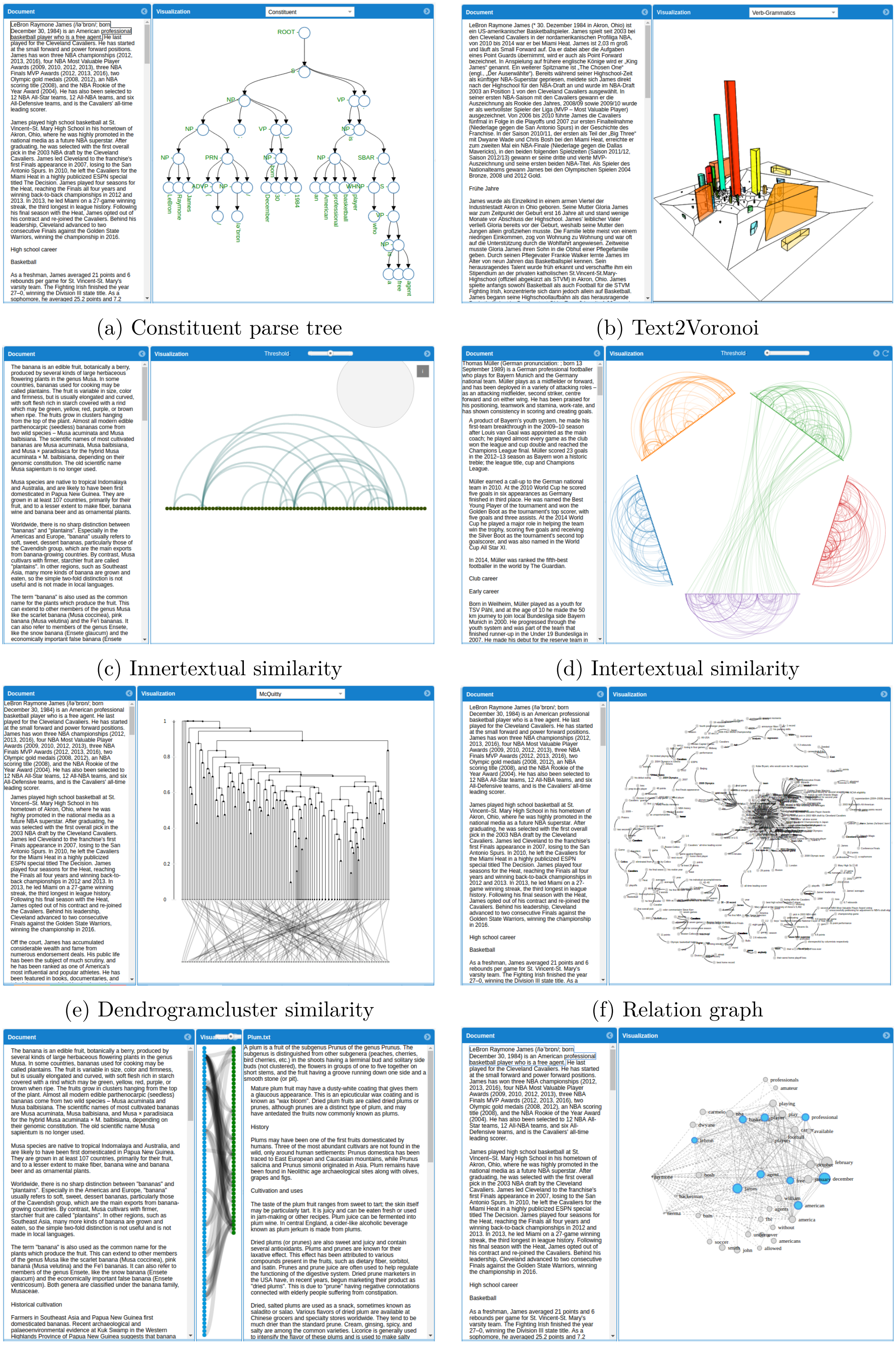
(b) LDA R-Module

Last but not least, TextImager provides a generic interface to *R*[5]. The aim is to give access to any NLP-related package in R *once more without requiring programming skills*. This is especially needed for scholars in digital humanities who are not trained in using script languages for modeling statistical procedures, but expect a versatile tool encapsulating this computational complexity. Thus, TextImager users can process input texts using R packages like LDA (see Figure 4.3b), network analysis or stylometrics (see Figure 4.3a) without the need to manipulate or to invoke any *R* script directly. All these R packages are given a single entrance point in the form of TextImager. See Mehler, Uslu, and Hemati (2016) for a recent research study based on TextImager.

4.1.5. Future Work

In already ongoing work, we extend the functionality of TextImager. This includes covering all features of tools like conTEXT. In contrast to many current frameworks, we will make TextImager's source code open-source as soon as the framework reaches a stable and documented version. We are going to specify a comprehensive model for component specification. The model will contain specifications of general components and their dependency hierarchy. This model will help defining where new NLP components are settled within the NLP landscape.

[5] `https://www.r-project.org`



(a) Constituent parse tree

(b) Text2Voronoi

(c) Innertextual similarity

(d) Intertextual similarity

(e) Dendrogramcluster similarity

(f) Relation graph

(g) Bipartite similarity

(h) Semantic relation graph

Figure 4.4.: Visualization Examples

4.1.6. Scope of the Software Demonstration

A beta version of TextImager's web application can be found at `http://textimager.hucompute.org`. A preprocessed demonstration can be found at `http://textimager.hucompute.org/index.html?viewport=demo`. A tutorial on how to set up TextImager's backend services on codebase and a list of available components and options can be found at `http://service.hucompute.org`.

Acknowledgment

We gratefully acknowledge financial support of this project via the BMBF Project CEDIFOR (`https://www.cedifor.de/en/`).

4.2. Evaluating and Integrating Databases in the Area of NLP

Hemati, Wahed, Alexander Mehler, Tolga Uslu, Daniel Baumartz, and Giuseppe Abrami (2018). “Evaluating and Integrating Databases in the Area of NLP”. In: *International Quantitative Linguistics Conference (QUALICO 2018)*. Wroclaw, Poland.

4.2.1. abstract

Since computational power is rapidly increasing, analyzing big data is getting more popular. This is exemplified by word embeddings producing huge index files of interrelated items. Another example is given by digital editions of corpora representing data on nested levels of text structuring. A third example relates to annotations of multimodal communication comprising nested and networked data of various (e.g., gestural or linguistic) modes. While the first example relates to graph-based models, the second one requires document models in the tradition of TEI whereas the third one combines both models. A central question is how to store and process such *big* and *diverse* data to support NLP and related routines in an efficient manner. In this paper, we evaluate six *Database Management Systems* as candidates for answering this question. This is done by regarding database operations in the context of six NLP routines. We show that none of the DBMS consistently works best. Rather, a family of them manifesting different database paradigms is required to cope with the need of processing big and divergent data. To this end, the paper introduces a web-based *multi-database management system* (MDBMS) as an interface to varieties of such databases.

4.2.2. Introduction

Digital humanities and related disciplines deal with a variety of data ranging from large index files (as in the case of word embeddings (Mikolov, Chen, et al. 2013)) to inclusion hierarchies as mapped by document models in the tradition of TEI (Burnard and Bauman 2007) and models of nested and networked data as exemplified by multimodal communication (Carletta et al. 2003). With the availability of web-based resources we observe both an increase concerning the size of such data – that is, in terms of big data – and the need to cope with this diversity simultaneously within the same project: lexica (represented as networks) are interrelated, for example, with large corpora of natural language texts (represented as tree-like structures) to observe language change (Kim et al. 2014; Michel et al. 2010; Eger and Mehler 2016). Another example is given by digital editions (Kuczera 2016) in which annotations of textual data are interrelated with annotations of pictorial representations of the underlying sources (Leydier et al. 2014; Lavrentiev, Stutzmann, and Leydier 2015). We may also think of text mining in the area of learning analytics based on big learning data, in which textual manifestations of task descriptions are interrelated with student assessments and qualitative data (Robinson et al. 2016; Crossley et al. 2015). To cope with such diversities, databases are required that allow for efficiently storing, retrieving and manipulating data ranging from distributions to tree-like structures and graphs of symbolic and numerical data. As a matter of fact, such an omnipotent database is still out of reach. Thus, the question is raised *which existing database technology based on which paradigm performs best in serving these tasks.* Alternatively, the question is raised *which combination of these technologies best fits these tasks.*

In this paper, we evaluate six *Database Management Systems* (DBMS) as candidates for answering these two questions. This is done by regarding elementary database operations that underlay typical scenarios in NLP or natural language annotation. We show that none of the DBMS under consideration consistently works best. Rather, families of such DBMS manifesting different database paradigms cope better with the rising needs of processing big as well as divergent data. To demonstrate this, the paper introduces a web-based *multi-database management system* (MDBMS) that allows for integrating families of databases. This is done to provide a single interface for querying different databases. To this end, the MDBMS encapsulates the specifics of the query languages of all databases involved. By using it, modelers are no longer forced to choose a single database, but can benefit from different databases that capture a variety of data modeling paradigms. This enables projects to simultaneously manage and query a variety of data by means of a multitude of databases – in a way that also allows for processing big data. Such a versatile MDBMS is especially interesting for NLP frameworks (Eckart de Castilho and Gurevych 2014; Hemati, Uslu, and Mehler 2016; E. W. Hinrichs, M. Hinrichs, and Zastrow 2010; OpenNLP 2010; Popel and Zabokrtský 2010) that still do not store annotations of documents in a query-able format. In cases where such annotations are stored according to the *UIMA Common Analysis Structure* (UIMA-CAS), serialization is based on XML-files. This makes operations such as collecting statistical information from documents very time-consuming, since one first has to deserialize each file involved. The same problem occurs when trying to add annotation layers. For additionally annotating, for example, named entities, one has two options: either one runs the underlying UIMA pipeline together with the named-entity recognition (NER) from scratch or the serialized output of the pipeline is deserialized and then made input to NER. Obviously, both variants are very time-consuming. A better option would be to let NER work only on those annotations that are needed for annotating named entities. However, this presupposes that all annotations are stored in a query-able format. By example of UIMA-CAS the present paper proposes a MDBMS that allows exactly this: to query annotations of documents according to varying data models even in cases where the amount of stored data ranges in terms of big data.

4.2.3. Related Work

For testing our framework, we include databases[6] that address different core tasks.

Until now, there has not been much research on the comparative evaluation of DBMS in the context of NLP. What is missing is a comparison of databases according to competing (relational, graph- or document-oriented) paradigms. Rather, databases are assessed in isolation. As an example for document-oriented databases, Richardet, Chappelier, and Telefont (2013) use MongoDB to provide

[6] `http://db-engines.com/en/ranking`

an interface for UIMA-based applications. Fette, Toepfer, and Puppe (2013) present a relational framework for storing CAS objects in a MySQL database. Another example is given by Hahn et al. (2008) who serialize UIMA-CAS objects by means of PostgreSQL. Since graph databases have become popular in digital humanities (Kuczera 2016), there are many examples according to this paradigm. Lyon (2016) use Neo4j to mine word associations. Ganesan, C. Zhai, and Han (2010) introduce a graph-based summarization framework. R. Mihalcea and Tarau (2004) exemplify a graph-based ranking model for text processing. Usbeck et al. (2014) use graph-based representations of texts and linked data for disambiguating named entities. Rousseau, Kiagias, and Vazirgiannis (2015) describe an approach to graph-based text classification in which texts are represented as graphs of words, while Tixier, Malliaros, and Vazirgiannis (2016) introduce a graph-based approach to keyword extraction. Finally, B. Müller and Hagelstein (2016) present an approach to analyzing and visualizing textual content that combines a graph database (Neo4j) with MongoDB. As enumerated in Section 4.2.4, we evaluate all of these and related databases by means of storing and retrieving annotations generated by six NLP operations.

4.2.4. Database Management Systems

This section describes the DBMS included to our evaluation. We use the UIMA type system of Eckart de Castilho and Gurevych (2014) as a reference to refer to annotation layers to be managed by the databases. Further, the evaluation utilizes TextImager (Hemati, Uslu, and Mehler 2016) to get access to a large variety of NLP routines as test cases. The valuation scenario refers to the annotation of text corpora along six annotation layers that are output by six selected routines. Any annotation information is stored by means of CAS objects to map between the UIMA type system on the one hand and the different DBMS on the other. In this section our question is then how the selected DBMS store input corpora and annotations. Section 4.2.5 will then evaluate how these annotations are managed by the DBMS.

Baseline Scenario: File System

To get a baseline scenario, we generate compressed files out of serialized UIMA-CAS objects output by NLP and store them in the file system. Serialization uses XMI[7], a standard for representing objects in XML (Grose, Doney, and Brodsky 2002). Note that for each operation at document level, any compressed file has to be de-compressed and -serialized. Obviously, this results in a memory overhead.

MongoDB

As an example of a scalable document-orientated NoSQL database we evaluate `MongoDB`.[8] To this end, we serialize UIMA-CAS objects into binary-encoded JSON Objects. Due to its document-oriented character, these objects can be added directly to MongoDB next to the type system. This makes it extremely flexible especially in the case of frequently changed type systems.

Because of being schema-free, MongoDB supports two ways for representing relations among objects: in terms of *embedded documents* or *document references*. Embedded documents store objects relations in a single Binary JSON document. To this end, input documents are denormalized and stored contiguously on disk. This results in better reading performance when querying entire documents, since only one query is required (Copeland 2013). However, document size can grow rapidly as each annotation layer is represented as a separate array of objects in the respective document. Since MongoDB has a size limit of 16 MB per document, storage of large documents is considerably limited in this scenario. Alternatively, relations can be stored by means of document references. In this case, one speaks of *normalized data* (Dayley 2014). In our evaluation of MongoDB, every annotation of a CAS object is stored as a separate JSON document. This allows for representing and storing larger CAS objects than in the case of embedded documents. Database size is further reduced by avoiding redundancy. A disad-

[7] `http://omg.org/spec/XMI/`

[8] `https://mongodb.com/`

vantage of normalized data is that multiple lookups are required when resolving dependencies and references (Dayley 2014).

Cassandra

`Apache Cassandra`[9] is a column oriented NoSQL database that is horizontally scalable. To this end, Cassandra replicates data to multiple nodes of a cluster to ensure fault tolerance (Hewitt 2011). For every annotation layer of the underlying type system we generate a separate table whose attributes correspond to those of the focal layer. Note that Cassandra does not allow for joining tables. From the point of view of the relational paradigm, this is problematic. Thus, dependency parses, semantic roles, anaphora and related relational data can hardly be modeled using Cassandra. However, an alternative is to distribute such data and synthesize it at runtime using query families.

MySQL

For evaluating `MySQL`[10], we use the NLP-related database schema of Fette, Toepfer, and Puppe (2013). It consists of six tables for storing annotations next to the type system.

BaseX

`BaseX`[11] is a light-weight XML document-oriented database using an XPath/XQuery 3.1 processor. In our setup, UIMA-CAS objects are serialized into XMI documents and added to this XML database. XPath/XQuery provides functionality for querying XMI representations of CASes in accordance with the type system.

[9] `https://cassandra.apache.org/`
[10] `https://mysql.com/`
[11] `http://basex.org/`

Neo4j

Neo4j[12] is a highly performant NoSQL graph database addressing networked data (J. J. Miller 2013). In our experiment, we represent a text corpus as an adjacency graph in Neo4j. That is, each input text is represented by a node linking to all its token nodes whose syntagmatic order is mapped by token links. Further, any annotation is represented by nodes in Neo4j so that no pair of different nodes instantiates the same attribute-value pair. That is, the attribute-value pair $(POS, verb)$, for example, is bijectively mapped to a node in Neo4j. This approach reduces the number of nodes, while raising the number of edges.

4.2.5. Experiments

We now evaluate the DBMS of Section 4.2.4 regarding the dual task of processing diverse as well as big data. To this end, we created a corpus of 70 000 German Wikipedia articles, each containing annotations of (1) *tokenization*, (2) *lemmatization*, (3) *POS tagging*, (4) *dependency parsing*, (5) *NER* and (6) *time recognition*. Table 4.1 lists the number of annotations generated for each of these layers. In the following sections, we evaluate the DBMS with respect to writing and reading the respective annotations. This evaluation is further differentiated by evaluating the performance of the DBMS with regard to querying attribute values and relational data.

Storage Performance

For each database of Section 4.2.4 we implemented a separate *UIMA Writer* to transform CAS objects output by the selected NLP routines to the CAS adaptations of the respective databases. Figure 4.5 shows the performance of each database in terms of single threaded write operations. Table 4.2 shows the time taken by the DBMS to store all documents. The time is measured in terms of serializing and storing CAS objects based on the respective database format.

[12] https://neo4j.com/

Table 4.1.: Number of annotations per annotation layer.

Layer	**Count**
Document	70 000
Paragraph	598 041
Sentence	1 520 441
Token	31 647 060
Character	169 131 822
POS	31 647 060
Lemma	31 647 060
Dependency	31 647 492
Named Entity	2 078 212
Time	1 039 106

Obviously, while file system-based storage is fastest, MySQL is worst. Table 4.2 shows usage of disk space induced by all inserts. While the file system is still best, the worst case is now BaseX.

Table 4.2.: Performance statistic for storing documents.

	XMI	**Mongo**	**Neo4j**	**BaseX**	**MySQL**	**Cas.**
Runtime (s)	1 190	4 953	4 227	17 101	27 283	5 858
Disk (GB)	4,8	15,8	22,6	33,0	28,4	10,8

Reading Performance

For each database we implemented a *UIMA CollectionReader* for mapping between CAS adaptations of DBMS and NLP routines. This mode concerns situation in which subsequent NLP routines operate on pre-annotated data. Figure 4.6 as well as Table 4.3 show the performance of the DBMS when performing single threaded read operations. Now, Neo4j outperforms all its competitors, while BaseX is still worst.

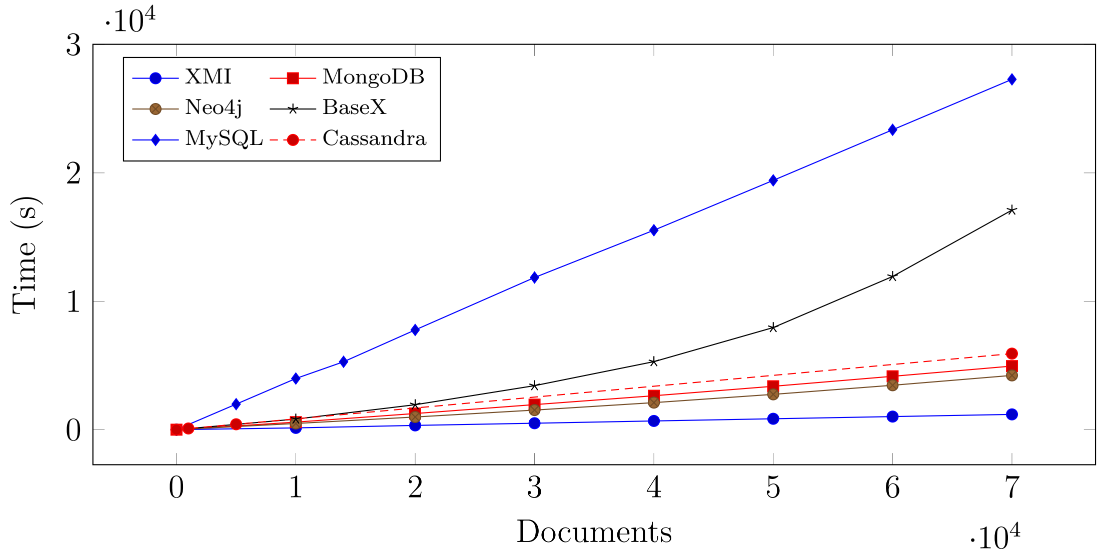


Figure 4.5.: Time of writing using the corresponding DBMS.

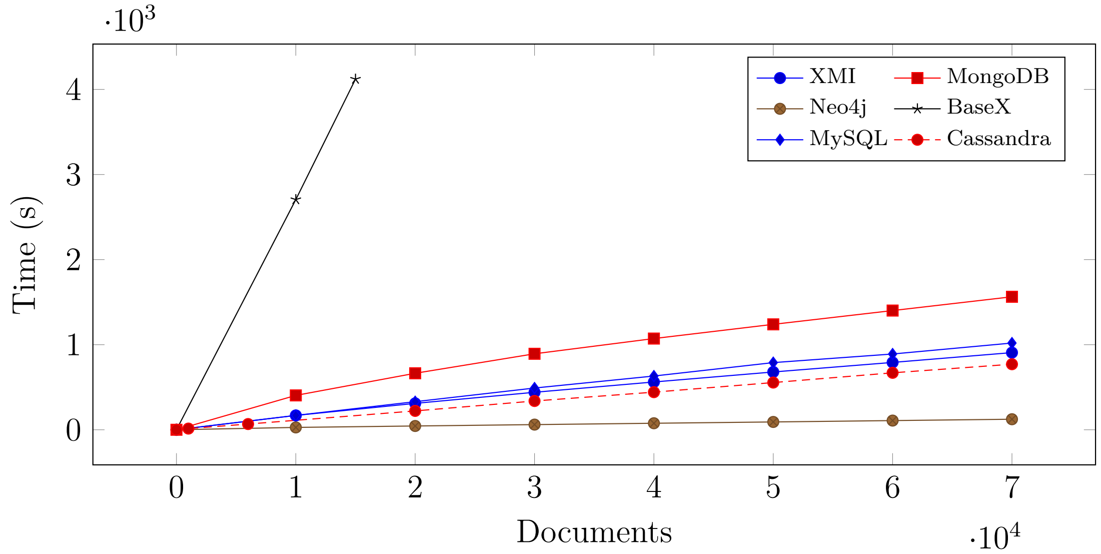


Figure 4.6.: Time need to read documents.

Table 4.3.: Time (in seconds) taken to read documents.

XMI	Mongo	Neo4j	BaseX	MySQL	Cas.
906 s	1 563 s	123 s	20 217 s	1 009 s	770 s

Query Performance

Databases typically provide expressive query languages. The more expressive this language, the more informative the data that can be queried, for example, for text mining, distant reading or text visualization. Such a query mode is particularly interesting for NLP frameworks (see Section 4.2.2) that still do not rely on query languages. In our experiment, we distinguish between querying attributes and relations. This is done by example of the task of generating frequency distributions for annotation layers.

Table 4.4.: Performance statistics of querying in seconds.

	XMI	Mongo	Neo4j	BaseX	MySQL	Cass.
POS	1 290,2	24,9	13,8	127,6	16,3	∅
Lemma	1 287,0	27,5	57,0	185,9	14,6	∅
Morph	1 299,1	26,3	48,0	132,3	18,6	∅
Dep	1 490,2	371,4	59,4	2 349,7	87,7	∅

Querying Attribute Values We query the databases for counting attributes like POS or lemma and use this data to calculate type-token ratios etc. Table 4.4 lists the time required by the DBMS to determine the corresponding frequency distributions. Note that Cassandra does not support count operations. The best performer is once more Neo4j, while the file system performs worst. Obviously, querying requires a DBMS.

Querying Relational Data For running a performance test regarding relations (e.g., dependency relations), we reconstructed the experiment of Fette, Toepfer, and Puppe (2013). That is, all dependency relations of all words are queried governing the verb “*gehen*”/“*to walk*”. Tokens of this lemma are retrieved together with all dominating or dependent tokens. Table 4.4 row *Dep* shows the test results: the best performer Neo4j, while BaseX is worst. Since Cassandra does not support join operations, we did not model dependency relations using this DBMS.

Combining databases

In this section we briefly describe how we combined databases to generate an MDBMS. The underlying optimization process focuses on two dimensions: runtime and memory consumption. Since the tasks described in Section 4.2.5 are independent of each other, the MDBMS is generated in such a way that it consists of databases that require a minimum runtime for the respective task. Thus, the best performing combination of databases is Neo4j (*read*,*dependency*), MySQL (*pos*,*lemma*,*morph*).

4.2.6. System Demonstration of MDBMS

In this section, we demonstrate the MDBMS as a framework for combining multiple DBMS. Each database to be integrated into the MDBMS has to be implemented as a Java interface. This interface provides functions for reading, writing and querying the respective database. To this end, we have written interfaces for all databases described in Section 4.2.4. In order to allow for using the MDBMS in a user-friendly manner, we utilize the web interface of TextImager (Hemati, Uslu, and Mehler 2016) (see Figure 4.7). TextImager is a UIMA-based framework that includes a wide range of NLP tools and visualization techniques for numerous languages. TextImager can upload and process multiple documents, compressed corpora and downloadable URLs according to its language-specific pipelines. All data preprocessed or output by TextImager is then stored in the MDBMS. Our system demonstration shows how TextImager uses the MDBMS to perform the tasks described in Section 4.2.5

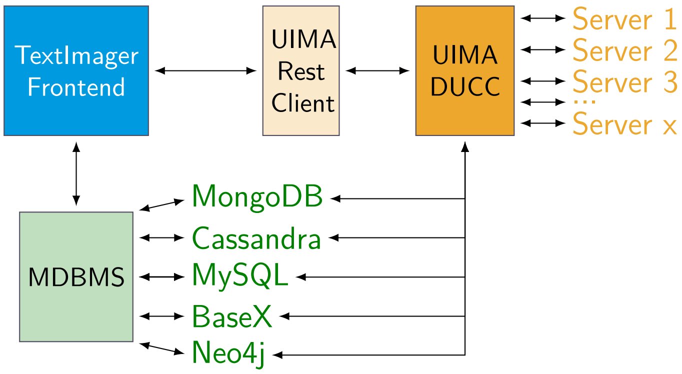


Figure 4.7.: The MDBMS as being interfaced by TextImager.

and to visualize the corresponding results. The demonstration can be found at `http://textimager.hucompute.org/lab/database/index.html`.

4.2.7. Conclusion and Future Work

We evaluated six DBMS in the context of six database operations underlying typical working scenarios in NLP. We showed that none of the DBMS under evaluation consistently works best, but rather a combination of databases addressing different paradigms of data modeling. That is, facing the rising needs of processing big as well as divergent data, families of such divergent DBMS are the better choice. We addressed this need by means of a so-called *Multi-DataBase Management system* (MDBMS) as an interface to families of such databases. Note that our evaluation of the DBMS in question considered independent tasks. In the case of non-circularly dependent ones, the optimization can be performed by means of a multistage graph G whose nodes denote combinations of tasks and databases. In this scenario, a node x is assigned to stage Y of smallest order, so that all tasks that are required to be completed before x belong to stages preceding Y. Edges connecting nodes of different stages represent dependencies of task completion whose weight equals the runtime required by completing the task denoted by the source node by means of the respective database. This allows for solving the underlying optimization problem by means of dynamic programming for minimizing cost of possible paths within G (Sniedovich 2010). This will be our task in future work.

4.3. Der TextImager als Front- und Backend für das verteilte NLP von Big Digital Humanities Data

Hemati, Wahed, Alexander Mehler, Tolga Uslu, and Giuseppe Abrami (2019). "Der TextImager als Front- und Backend für das verteilte NLP von Big Digital Humanities Data". In: *Proceedings of the 6th Digital Humanities Conference in the German-speaking Countries, DHd 2019*. DHd 2019. Frankfurt, Germany.

4.3.1. abstract

Immer mehr Disziplinen benötigen *Natural Language Processing* (NLP) Werkzeuge, um automatische Textanalysen auf verschiedenen Ebenen der Sprache durchzuführen. Die Anzahl der NLP-Werkzeuge wächst rasant[13]. Auch die Anzahl der frei oder anderweitig zugänglichen Ressourcen wächst. Angesichts dieser wachsenden Zahl an Werkzeugen und Ressourcen ist es schwierig, den Überblick zu behalten; gleichzeitig ist ein *Computational-Linguistic*-Framework, das große Datenmengen aus verschiedenen Quellen verarbeiten kann, noch nicht etabliert. Ein solches Framework sollte in der Lage sein, Daten verteilt zu verarbeiten und gleichzeitig eine standardisierte Programmier- und Modellschnittstelle bereitzustellen. Darüber hinaus sollte es modular und leicht erweiterbar sein, um die ständig wachsende Palette neuer Ressourcen und Tools zu integrieren. Das Framework muss offen genug für Erweiterungen Dritter sein, wobei jede Erweiterung für die gesamte Community zugänglich bleibt. Das Framework sollte es zudem Dritten ermöglichen, den Zugang zu ihren Erweiterungen zu beschränken, wenn dies beispielsweise durch Urheberrecht, geistiges Eigentum oder Datenschutz erforderlich ist. Um diesen Anforderungen gerecht zu werden, haben wir den TextImager (Hemati, Uslu, and Mehler 2016) um ein verteiltes Serversystem mit Cluster-Computing-Funktionen auf der Basis von UIMA (Ferrucci and Lally 2004) weiterentwickelt.

UIMA ist ein Framework zur Verwaltung von Datenflüssen zwischen Komponenten. Es bietet standardisierte Interfaces zur Erstellung von Komponenten

[13] https://github.com/topics/nlp

an. Dabei können die Komponenten einzeln oder im Verbund in einer Pipeline-Struktur ausgeführt werden. UIMA bietet weitgehende Möglichkeiten der sequenziellen Ordnung von NLP-Werkzeugen und verspricht, auch in Zukunft von der Community weiterentwickelt zu werden: Prozess-Management auf der Basis von UIMA erscheint nach derzeitigem Stand daher als erste Wahl im Bereich von NLP und DH.

TextImager bietet eine Vielzahl von UIMA-basierten NLP-Komponenten an, darunter unter anderen einen *Tokenizierer*, einen *Lemmatisierer*, einen *Part-Of-Speech-Tagger*, einen *Named-Entity-Parser* und einen *Dependency Parser*, und zwar für eine Vielzahl von Sprachen, darunter Deutsch, Englisch, Französisch und Spanisch. Dieses Spektrum an Werkzeugen besteht allerdings nicht ausschließlich aus Eigenentwicklungen, sondern wird maßgeblich um Entwicklungen Dritter erweitert, wozu unter anderem die Tool-Palette von *Stanford CoreNLP* (Manning et al. 2014), *OpenNLP* (OpenNLP 2010) und *DKpro* (Eckart de Castilho and Gurevych 2014) zählen.

In Zeiten von *Big Data* wird es immer relevanter, Daten schnell zu verarbeiten. Aus diesem Grund ist TextImager als Multi-Server- und zugleich als Multi-Instanz-Cluster aufgebaut, um das verteilte Verarbeiten von Daten zu ermöglichen. Dafür setzt TextImager auf UIMAs Cluster-Management-Dienste UIMA-AS[14] und UIMA-DUCC[15] auf.

Abbildung 4.8 zeigt eine schematische Darstellung von TextImager. Jede NLP-Komponente läuft als UIMA-AS Webservice auf dem Computing-Cluster des TextImager. Dabei können mehrere Instanzen einer Komponente instanziiert (s. Abbildung 4.8, *Service Instances*) werden und dennoch über eine Webservice-Schnittstelle (s. Abbildung 4.8, *UIMA AS Services*) angesprochen werden. Dazu wird das Java Messaging Service (JMS) verwendet, das die Kommunikation zwischen verschiedenen Komponenten einer verteilten Anwendung ermöglicht. JMS implementiert ein Point-to-Point-Kommunikationssystem. Dieser Kommunikationstyp basiert auf dem Konzept der *message queues* (Warteschlangen), *senders*

[14] `https://uima.apache.org/doc-uimaas-what.html`

[15] `https://uima.apache.org/doc-uimaducc-whatitam.html`

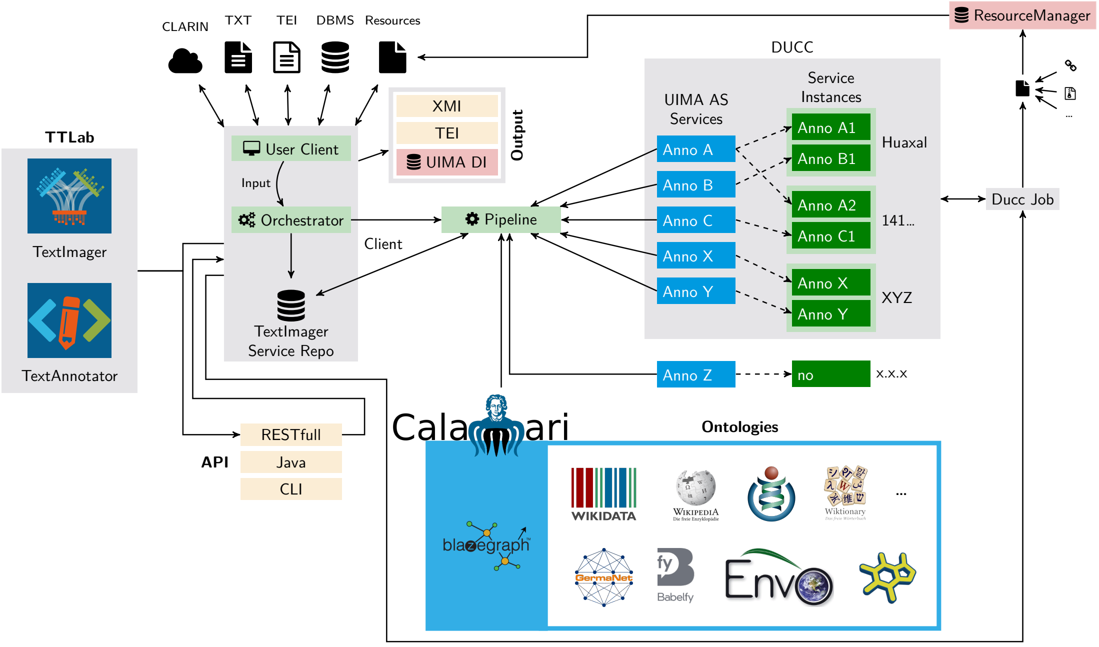


Figure 4.8.: TEXTIMAGER: system components and their relations.

(Sender) und *receivers* (Empfänger). Jedem Dienst ist eine Eingabewarteschlange und eine Ausgabewarteschlange zugeordnet. Um mehrere Instanzen einer Komponente zu verteilen, verbinden sich die Instanzen mit der gleichen Service-Eingangswarteschlange. Die Instanzen erhalten aus dieser Warteschlange Arbeitseinheiten. Nach der Verarbeitung wird das Ergebnis an eine Ausgabewarteschlange zurückgegeben. Die Ausgabewarteschlange eines Dienstes kann an eine Eingabewarteschlange eines anderen Dienstes angeschlossen werden, um eine Pipeline zu erstellen. Aufgrund dieser Ein- und Ausgabewarteschlangen-Systematik kann jeder Service Arbeitseinheiten asynchron bearbeiten. Durch diese Architektur ist TEXTIMAGER eine Multi-Server-, Multi-Service- und Multi-Service-Instanz-Architektur.

Darüber hinaus bietet TEXTIMAGER ein Toolkit, das es jedem Entwickler ermöglicht, einen eigenen TEXTIMAGER-Cluster aufzusetzen und Services im TEXTIMAGER-System hinzuzufügen. Entwickler können den Zugriff auf die Dienste einschränken, wenn dies wie oben beschrieben erforderlich ist, was mittels die Integration des ResourceManagers (Gleim, Mehler, and Ernst 2012) und des AuthorityManagers (Gleim, Mehler, and Ernst 2012) realisiert wird.

Durch Freigabe des Quellcodes des TextImager und die Bereitstellung von Leitlinien für dessen Erweiterung wollen wir es Dritten ermöglichen, ihre NLP-Software über die Webservices von TextImager zu vertreiben, so dass die gesamte wissenschaftliche Gemeinschaft davon profitiert. Installationsanweisungen und Beispiele für die Einrichtung eines TextImager-Servers finden Nutzer in folgendem GitHub-Repository: `https://github.com/texttechnologylab/textimager-server`.

Der Beitrag erörtert die Möglichkeiten und Grenzen des NLP von Big Data, stellt den TextImager als Werkzeug für diesen Bereich zur Diskussion und zeigt anhand von drei Nutzungsszenarien Einsatzmöglichkeiten in den DH auf.

4.4. TextImager as an interface to BeCalm

Hemati, Wahed, Tolga Uslu, and Alexander Mehler (2017). “TextImager as an interface to BeCalm”. In: *BioCreative V.5. Proceedings.*

4.4.1. abstract

TIPS (Technical interoperability and performance of annotation servers) is a novel BioCreative task. It focuses on the technical aspects of making NER taggers available as webservices. In this paper, we present the functionality and architecture of BeCalm metaserver integrated into TextImager. We present the integration of in-house developed NER tagger and also the integration and adaption of available NER tagger.

4.4.2. Introduction

The novel BioCreative task TIPS focuses on making NER taggers available as web servers. We introduced TextImager in Hemati, Uslu, and Mehler (2016) where we focused on the architecture and the functions of its backend. In this paper, we present TextImager as an annotation server on the BeCalm platform. TextImager is a UIMA(Ferrucci and Lally 2004)-Based framework that offers a range of NLP. It is modular and expendable, due to the underlying UIMA architecture. We made TextImager available online. It can be accessed by BeCalm metaserver requests and is able to answer such requests in the BeCalm TSV format. Currently we trained and implemented 6 NER systems for the BeCalm GPRO and CEMP tasks, namely StanfordNER, MarMot, CRF++, MITIE, Glample and CRFVoter. Each of these tagger can be used in a pipeline together with other NLP-tools that are already integrated in TextImager.

4.4.3. Systems description and methods

In this section we will describe the technical implementation of BeCalm metaserver integration into TextImager. We implemented a REST API that provides access and responds to the BeCalm metaserver requests described in `http://www.becalm.eu/api`

In order to obtain annotation documents the BeCalm metaserver sends a *getAnnotations()* request for a set of documents. The getAnnotations() message is in JSON format, which is exemplified in Listing 4.1. The request contains specification about the documents that need to be processed, authentication information and the *costum_parameter annotator*.

Listing 4.1: Example BeCalm metaserver JSON request

```
{
    ”method”:”getAnnotations”,
    ”becalm_key”:”141xxx”,
    ”name”:”becalm”,
    ”custom_parameters”:{
        ”annotator”:”BioGproS”
    },
    ”parameters”:{
        ”expired”:”2017–05–02T15:10:00+02:00”,
        ”documents”:[
            {
                ”source”:”ABSTRACT SERVER”,
                ”document_id”:”12140745”
            },

            {
                ”source”:”PATENT SERVER”,
                ”document_id”:”EP0959896B1”
            },
            ...
        ],
        ”communication_id”:”4672794”
    }
}
```

After receiving the *getAnnotations()* request, our server responds to BeCalm with an acknowledge message.

After the acknowledgement, our server executes the internal workflow, which starts by processing the BeCalm JSON request. The retrieved documents are downloaded by the TextImager Rest API from their specific sources, namely *Abstract Server*, *Patent Server* and *PubMed Server*. The documents are then passed to the TextImager backend. TextImagers backend is a UIMA-based framework that offers a range of NLP tools. Every TextImager component is configured as a UIMA-AS service, which may run standalone or in a pipeline (see Figure 4.9, *Stanford NER service Interface*, *MarMot Service Interface*, etc). All service instances are located on servers. Note that these instances can be distributed among different servers (see Figure 4.9, *Server 1, Server 2, Server X*). We trained and integrated 6 NER systems for the `BeCalm GPRO` and `CEMP` tasks into TextImager, namely `StanfordNER`(Finkel, Grenager, and Manning 2005), `MarMot`(T. Müller, Schmid, and Schütze 2013), `CRF++`(Kudo 2005), `MITIE`(Geyer et al. 2016), `Glample`(Lample et al. 2016) and `CRFVoter`.

The TextImager Rest API sends the request and the documents to the TextImager Orchestrator. The *custom_parameter: annotator* (see Listing 4.1) is used by the orchestrator to determine which NER service to run. The TextImager Orchestrator selects and acquires components and their resources: it arranges components into pipelines and grants the ability to parallelize them. Each of the 6 integrated NER systems require tokenization, lemmatization and morph tagging as preprocessing steps. Finally after the orchestration step, the documents are processed by means of the selected service pipeline. After processing is done, the output is passed to the TextImager Rest API, which saves the output to the BeCalm server, by calling the method *saveAnnotations*[16].

[16] `http://www.becalm.eu/api`

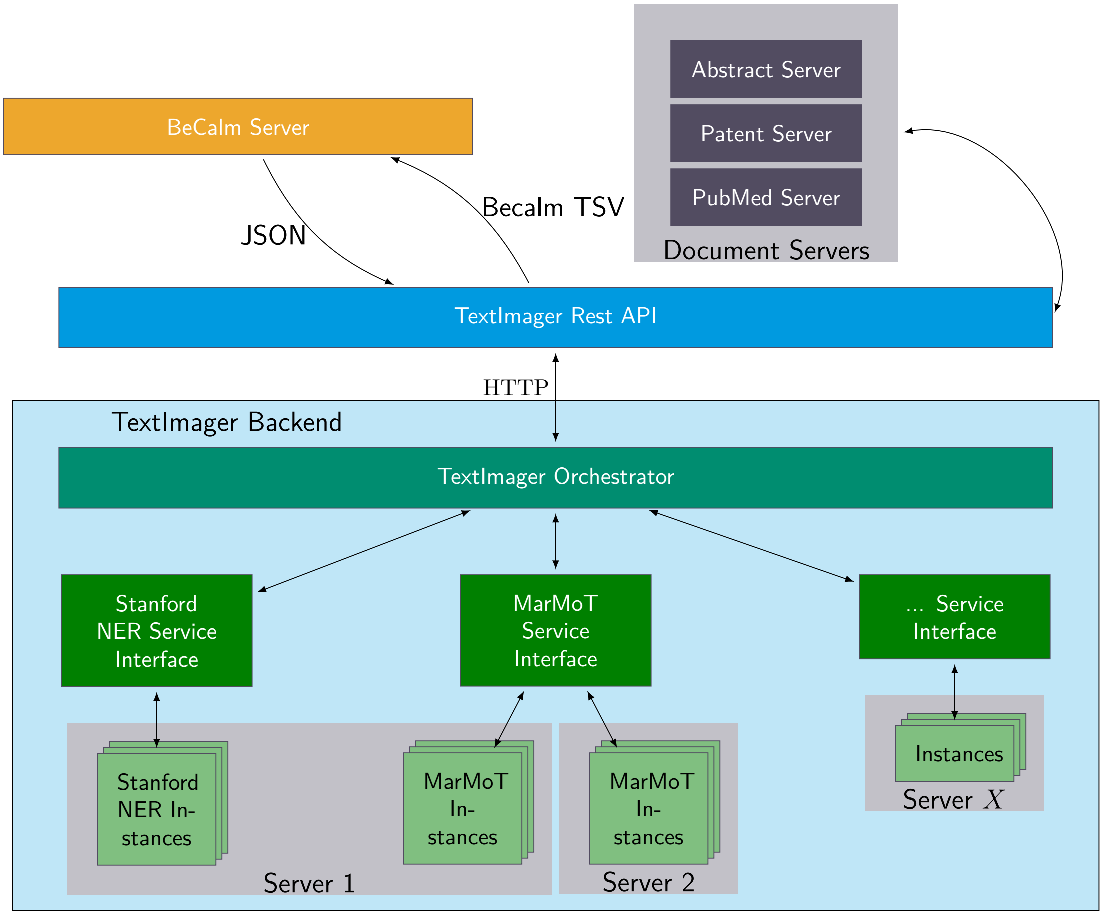


Figure 4.9.: Architecture of BeCalm metaserver integrated into TextImager

4.4.4. Discussion

In this work, we presented TextImager as a Becalm annotation server. Our annotation server is a distributed and modular framework. We currently integrated 6 NER systems for CEMP and GPRO. In future work, we will focus on integrating more NER systems into TextImager and make it available as annotation server. We will also provide more output formats.

Acknowledgments.

We gratefully acknowledge support by the *Deutsche Forschungsgemeinschaft* via the Specialised Information Services *Biodiversity Research*.

4.5. LSTMVoter: chemical named entity recognition using a conglomerate of sequence labeling tools

Hemati, Wahed and Alexander Mehler (2019b). "LSTMVoter: chemical named entity recognition using a conglomerate of sequence labeling tools". In: *J. Cheminformatics* 11.1, 3:1–3:7. DOI: `10.1186/s13321-018-0327-2`. URL: `https://doi.org/10.1186/s13321-018-0327-2`.

4.5.1. abstract

Background Chemical and biomedical *Named Entity Recognition* (NER) is an essential preprocessing task in *Natural Language Processing* (NLP). The identification and extraction of named entities from scientific articles is also attracting increasing interest in many scientific disciplines. Locating chemical named entities in the literature is an essential step in chemical text mining pipelines for identifying chemical mentions, their properties, and relations as discussed in the literature. In this work, we describe an approach to the BioCreative V.5 challenge regarding the recognition and classification of chemical named entities. For this purpose, we transform the task of NER into a sequence labeling problem. We present a series of sequence labeling systems that we used, adapted and optimized in our experiments for solving this task. To this end, we experiment with hyperparameter optimization. Finally, we present `LSTMVoter`, a two-stage application of *Recurrent Neural Network*s (RNN) that integrates the optimized sequence labelers from our study into a single ensemble classifier.

Results We introduce `LSTMVoter`, a bidirectional *Long Short-Term Memory* (LSTM) tagger that uses a conditional random field layer in conjunction with attention-based feature modeling. Our approach explores information about features that is modeled by an attention mechanism. `LSTMVoter` outperforms each extractor integrated by it in a series of experiments. On the BioCreative IV chemical compound and drug name recognition (CHEMDNER) corpus, `LSTMVoter` achieves an F1-score of 90,04 %; on the BioCreative V.5 chemical entity mention

in patents (CEMP) corpus, it achieves an F1-score of 89,01 %.

Availability and implementation Data and code are available at `https://github.com/texttechnologylab/LSTMVoter`

4.5.2. Introduction

In order to advance the fields of biological, chemical and biomedical research, it is important to stay on the cutting edge of research. However, given the rapid development of the disciplines involved, this is difficult, as numerous new publications appear daily in biomedical journals. In order to avoid repetition and to contribute at least at the level of current research, researchers rely on published information to inform themselves about the latest research developments. There is therefore a growing interest in improved access to information on biological, chemical and biomedical data described in scientific articles, patents or health agency reports. In this context, improved access to chemical and drug name mentions in document repositories is of particular interest: it is these entity types that are most often searched for in the PubMed (*PubMed - NCBI* n.d.) database. To achieve this goal, a fundamental preprocessing step is to automatically identify biological and chemical mentions in the underlying documents. Based on this identification, downstream NLP tasks such as the recognition of interactions between drugs and proteins, of side effects of chemical compounds and their associations with toxicological endpoints or the investigation of information on metabolic reactions can be carried out.

For these reasons, NLP initiatives have been launched in recent years to address the challenges of identifying biological, chemical and biomedical entities. One of these initiatives is the BioCreative series, which focuses on biomedical text mining. BioCreative is a “Challenge Evaluation”, in which the participants are given defined text mining or information extraction tasks in the biomedical and chemical field. These tasks include *Gene Mention detection (GM)* (Smith et al. 2008; Hirschman et al. 2005), *Gene Normalization (GN)* (Hirschman et al. 2005; Morgan et al. 2008; Z. Lu et al. 2011), *Protein-Protein Interaction (PPI)*

(Krallinger, Vazquez, et al. 2011), *Chemical Compound and Drug Name Recognition (CHEMDNER)* (Krallinger, Leitner, et al. 2015; Krallinger, Rabal, et al. 2015) and *Chemical Disease Relation Extraction* (Li et al. 2016; Wei et al. 2016) tasks.

The current *BioCreative V.5* task consists of two off-line tasks, namely *Chemical Entity Mention in Patents (CEMP)* and *Gene and Protein Related Object Recognition (GPRO)*. CEMP requires the detection of chemical named entity mentions. The task requires detecting the start and end indices corresponding to chemical entities. The GPRO task requires identifying mentions of gene and protein related objects in patent titles and abstracts (Krallinger, Martin Pérez-Pérez, et al. 2017). In this work, we focus on the CEMP task. The CEMP task is an abstraction of the common Named Entity Recognition (NER) tasks, which can be reduced to a sequence labeling problem, where the sentences are represented as sequences of tokens. The task is then to tag chemical entity mentions in these sequences. The settings of the CEMP task are similar to the Chemical Entity Mention Recognition (CEM) subtask of CHEMDNER challenge in BioCreative IV (Krallinger, Leitner, et al. 2015). Therefore, we addressed both tasks and their underlying corpora in our experiments. Note that the current article describes an extension of previous work (Hemati, Mehler, and Uslu 2017).

The article is organized as follows: First we describe our methodical apparatus and resources. This includes the data and corpora used in our experiments. Then, we introduce state-of-the-art tools for NER and explain how we adapted them to perform the CEMP task. Next, we present a novel tool for combining NER tools, that is, the so-called `LSTMVoter`. Finally, we present our results, conclude and discuss further work.

4.5.3. Materials and Methods

In this section, we first describe the datasets used in our experiments. Then, the two-stage application of `LSTMVoter` is introduced.

Datasets

In our experiments, two corpora of the BioCreative Challenge were used: the CHEMDNER Corpus (Krallinger, Leitner, et al. 2015) and the CEMP Corpus (M Pérez-Pérez et al. 2017).

The CHEMDNER corpus consists of 10 000 abstracts of chemistry-related journals published in 2013. Each abstract was human annotated for chemical mentions. The mentions were assigned to one of seven different subtypes (ABBREVIATION, FAMILY, FORMULA, IDENTIFIER, MULTIPLE, SYSTEMATIC, and TRIVIAL). The BioCreative organizer divided the corpus into training (3 500 abstracts), development (3 500 abstracts) and test (3 000 abstracts) sets.

For CEMP task, the organizers of *BioCreative V.5* provided a corpus of 30 000 patent abstracts from patents published between 2005 and 2014. These abstracts are divided into training (21 000 abstracts) and test (9 000 abstracts) sets. The corpus is manually annotated with chemical mentions. For the construction of the CEMP corpus the annotation guidelines of CHEMDNER were used. Therefore, CEMP contains the same seven chemical mention subtypes as CHEMDNER. Table 4.5 shows the number of instances for both corpora for each of these subtypes.

Table 4.5.: Number of instances for each subtype of CEMP and CHEMDNER corpus.

Annotation	**CEMP**	**CHEMDNER**
ABBREVIATION	1 373	9 059
FAMILY	36 238	8 313
FORMULA	6 818	8 585
IDENTIFIER	278	1 311
MULTIPLE	418	390
SYSTEMATIC	28 580	13 472
TRIVIAL	25 927	17 802
NO CLASS	0	72
Total count	99 632	59 004

Both corpora were enriched with additional linguistic features. For this, multiple preprocessing steps were applied on each set including sentence splitting, tokenization, lemmatization and fine-grained morphological tagging by means of Stanford CoreNLP (Manning et al. 2014). In addition, tokens were split on non-alphanumeric characters, as this variant brought a performance increase. Since the chemical mention detection task can be reduced to a sequence labeling problem, the corpora were converted into a sequence structure. To this end, a sequence of documents with sequences of sentences each containing a sequence of tokens was constructed and transformed according to a TSV format. Each word and its associated features are in one line separated by tabs. Sentences are separated by an empty line. For the labeling of the mentions, the IOB tagging scheme (Ramshaw and Marcus 1995a) was used (I = *inside of an entity*, O = *outside of an entity*, B = *beginning of an entity*). IOB allows the annotation of entities that span multiple tokens, where the beginning and the end of the entity is marked. This enables models to learn transition probability. `LSTMVoter` needs four datasets for the training process. Two pairs of training and development sets are required. Each pair is needed in one of the two stages of `LSTMVoter` (see Section *System Description*). Therefore, we divided the training set of CEMP into two series of training, development and test sets (each half of the original training set was split according to the pattern 60 %/20 %/20 %), where the first series is used for stage one, and the second for stage two. For the CHEMDNER corpus the available training and development sets were joined and split into training and development sets according to the schema 80 %/20 % – as before, we distinguish two such series. For evaluating our classifiers with respect to CHEMDNER, the test set provided by the organizers of the challenge was used. For the following experiments we used the corpora described as so far.

System Description

In this section we describe our system. Our approach implements a two-stage application of Long short-term memory (LSTM) using a conglomerate of sequence labelers for the detection of chemical mentions.

In the first stage, we trained and optimized five tools for NER for tackling this task, namely *Stanford Named Entity Recognizer* (Finkel, Grenager, and Manning 2005), *MarMoT* (T. Müller, Schmid, and Schütze 2013), *CRF++* (Kudo 2005), *MITIE* (Geyer et al. 2016) and *Glample* (Lample et al. 2016). For each of them, we optimized the corresponding hyperparameter settings. Generally speaking, hyperparameter tuning is a challenging task in machine learning. The optimal set of hyperparameters depends on the model, the dataset and the domain (Claesen and Moor 2015). Our experiments focused on optimizing the hyperparameters of each NER system independently, which led to a noticeable increase in F-score compared to the default settings. For each NER, we performed the Tree-structured Parzen Estimator (TPE) (Bergstra, Bardenet, et al. 2011) with 200 iterations. The results of the best performing model for each of these NER is listed in Table 4.6.

Table 4.6.: Comparison of annotators trained and tested on CEMP and CHEMDNER corpora measured by precision (P), recall (R), f1-score (F1)

.

System	**CEMP**			**CHEMDNER**		
	P	**R**	**F**	**P**	**R**	**F**
Stanford NER	0,85	0,80	0,82	0,82	0,83	0,82
MarMoT	0,87	0,86	**0,86**	0,85	0,85	**0,85**
CRF++	0,77	0,73	0,73	0,74	0,71	0,73
MITIE	0,65	0,65	0,65	0,62	0,61	0,62
Glample	0,76	0,79	0,77	0,82	0,84	0,83
Majority Vote	0,78	0,79	0,78	0,70	0,76	0,73
`LSTMVoter`	0,90	0,88	**0,89**	0,91	0,90	**0,90**

The NER tools are more or less independent of each other in the sense that one can find a subset of test cases that are correctly processed by one of them, but not by another. Therefore, combining these NERs is a promising candidate for increasing performance. We started with computing combinations of these NERs by means of a simple majority vote (Dietterich 2000), where the target label is selected, that is assigned by the majority of classifiers. Our experiments show that a simple majority vote brings no gain in performance compared to the best performing reference systems being examined in our study (see Table 4.6).

Thus, we developed a two-stage model, the so-called `LSTMVoter`, which trains a Recurrent Neural Network (RNN) with attention mechanism to learn the best combination of the underlying sequence labeling tools from stage one.

In the second stage, we combine the sequence labelers of stage one with two bidirectional *Long Short-Term Memory* (LSTM) networks with attention mechanism and a Conditional Random Field (CRF) network to form `LSTMVoter`. The architecture of `LSTMVoter` is illustrated in Figure 4.10. The core of `LSTMVoter` is based on Lample et al. (2016).

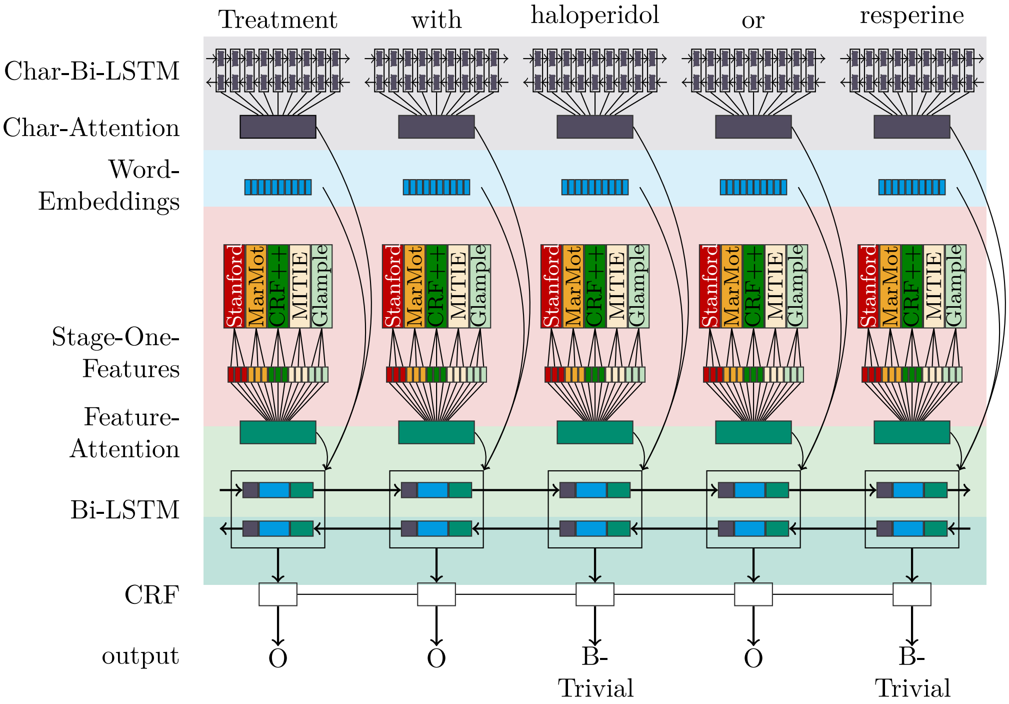


Figure 4.10.: Architecture of `LSTMVoter`.

LSTM networks are a type of RNN (Elman 1990). RNN allow the computation of fixed-size vector representations for sequences of arbitrary length. An RNN is, so to speak, a function that reads an input sequence $x_1, ..., x_n$ of length n and produces an output vector h_n, which depends on the entire input sequence. Though, in theory, an RNN is capable of capturing long-distance dependencies in the input sequence, in practice, they may fail due to the problem of vanishing gradients (Hochreiter 1998; Pascanu, Mikolov, and Bengio 2012). On the other hand, LSTMs include a memory cell, which can maintain information in memory for long periods of time (Hochreiter and Schmidhuber 1997; Hammerton 2003). This enables finding and exploiting long range dependencies in the input sequences to cope with the problem of vanishing gradients. Figure 4.11 illustrates an LSTM memory cell, which is implemented as follows:

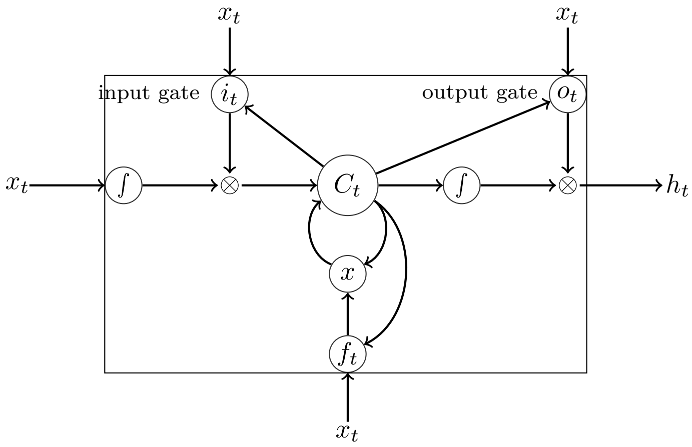


Figure 4.11.: A Long Short-Term Memory Cell.

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t \tanh(c_t)
\end{aligned}
$$

where x_t is the input vector (e.g. word embedding) at time t. h_t is the hidden state vector, also called output vector, that contains information at time t and all time steps before t. σ is the logistic sigmoid function (Weisstein 2002). Input gate i, forget gate f, output gate o and cell vector c are of the same size as the hidden state vector h. W_{hi}, W_{hf}, W_{hc} and W_{ho} are the weight matrices for the hidden state h_t. W_{xi}, W_{xf}, W_{xc} and W_{xo} denote the weight matrices of different gates for input x_t.

For LSTMVoter, we apply an LSTM to sequence tagging. Additionally, as proposed by Graves, Mohamed, and Hinton (2013), we utilize bidirectional LSTM networks. Figure 4.12 illustrates a bidirectional Long short-term memory (Bi-LSTM) network, where the input sequence (*Treatment with haloperidol or reserpine ...*) and the output sequence (*O, O, B-Trivial, O, B-Trivial, ...*) are fed as a training instance to a Bi-LSTM. In Bi-LSTMs, the input sequence is presented forward and backward to two separate hidden states to capture past and future information. To efficiently make use of past features (via forward states) and future features (via backward states) for a specific time frame, the two hidden states are concatenated to form the final output. In the final output of a Bi-LSTM, all information of the complete sequence is compressed into a fixed-length hidden state vector, which may result in information loss. To overcome this information loss, an attention mechanism is introduced, which partially fixes the problem.

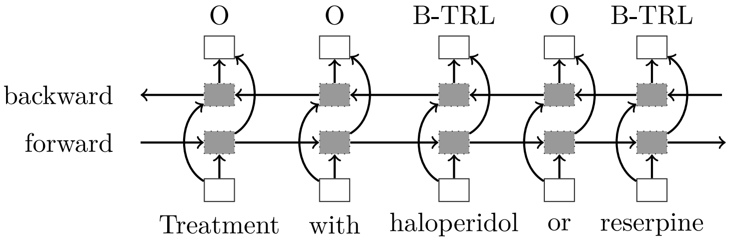


Figure 4.12.: A bidirectional LSTM network.

The method of attention mechanism has recently gained popularity in image caption generation (K. Xu et al. 2015), visual question answering (Zichao Yang et al. 2015) and language modeling tasks (Golub and X. He 2016; Rei, Crichton, and Pyysalo 2016; Luong, Pham, and Manning 2015; Bahdanau, Cho, and Bengio 2015). The attention mechanism plugs a context vector on top of a layer, which enables to take all cells' outputs as input to compute a probability distribution. This enables to capture global information rather then to infer based on one output vector.

For LSTMVoter, we utilized Bi-LSTM with attention mechanism to model character-level features (see Figure 4.10, *Char-Bi-LSTM*). Character-level features in chemical named entities contain rich structure information, such as prefix, suffix and n-grams. Unlike previous methods (Y. Lu et al. 2015; Khabsa and Giles 2015; S. Xu et al. 2015), character-level features do not have to be defined manually, rather they can be learned during training. Unlike Lample et al. (2016), who encodes the entire character sequence into a fixed-size vector for each word, we utilize the character-level attention mechanism introduced by Rei, Crichton, and Pyysalo (2016). This has the advantage, that by using the attention mechanism, the model is able to dynamically decide how much information and which part of a token to use.

In addition to the character-level features, we implemented word embeddings into our model to capture dependencies between words (see Figure 4.10, *Word-Embeddings*). For this, we evaluated various methods, namely GloVe (Pennington, Socher, and Manning 2014), Dependency-Based embeddings (Levy and Goldberg 2014; Komninos and Manandhar 2016) trained on the English Wikipedia, and word2vec (Mikolov, Sutskever, et al. 2013) trained on the English Wikipedia and a biomedical scientific literature corpus containing PubMed abstracts and full texts. In our experiments, the word2vec model trained on biomedical scientific literature gave the best results.

To utilize the results of the NERs from stage one, we encode the respective results of the NERs into one-hot vectors concatenated to a feature vector (see Figure 4.10, *Stage-One-Features*). An attention mechanism is placed on the feature vector. By creating a probability distribution through the attention mechanism, `LSTMVoter` learns how to weight each result of the NERs from stage one. With the attention vector it is even possible to determine for each element of a sequence how important the individual partial results from stage one were. This has the advantage that the model is no longer a black box, but can be interpreted as to how important the individual results from stage one were.

All previous elements of `LSTMVoter` encode word-based information. Another Bi-LSTM is used to learn relationships between these word-based information (see Figure 4.10, *Bi-LSTM*).

To deal with the independent label output problem, we utilize the output vector as elements. For this we combine the Bi-LSTM layer with a linear-chain CRF (see Figure 4.10, *CRF*). Linear-chain CRFs define the conditional probability of a state sequence to be:

$$P(y|x) = \frac{1}{Z_x} exp \left(\sum_{j=1}^{n} \sum_{m=1}^{l} \lambda_m f_m(y_{j-1}, y_j, x, j) \right)$$

where Z_x is the normalization factor that makes the probability of all state sequences sum to one; $f_m(y_{j-1}, y_j, x, j)$ is a feature function, and λ_m is a learned weight associated with feature f_m. Feature functions measure the aspect of a state transition, $y_{j-1}, y_j \rightarrow y_t$, and the entire observation sequence, x, centered at the current time step, j. Large positive values for λ_m indicate a preference for such an event, whereas large negative values make the event unlikely.

Finally, to optimize the hyperparameters, the Tree Structure Parzen estimator was used.

4.5.4. Results

This section presents the results of our experiments for the chemical named entity recognition on CEMP and CHEMDNER corpus. For evaluation the BioCreative Team has specified standard evaluation statistics, namely precision (P), recall (R) and F1-score (F) (M Pérez-Pérez et al. 2017). For each sequence labeling tool, the hyperparameters were optimized using Tree Structure Parzen Estimators, which led to a noticeable increase of performance. For example, in the optimization process of CRF++, the difference between the worst to the best performer is 65%. The results show the need for machine learning algorithms to perform hyperparameter optimization.

Table 4.6 shows the comparison of annotators trained on CEMP and CHEMDNER corpus. The results listed are those obtained after the hyperparameter optimization described in the Methods Section, which were trained, optimized and tested on the corpora described in this Section. Each sequence labeling system classifies a different subset correctly. The combination of sequence labelling systems in a majority vote did not improve performance and is even below the best sequence labelling systems. In contrast, `LSTMVoter` increases the performance and performs best in our experiments.

4.5.5. Conclusions

In this work, we compared a set of sequence labeling systems. We trained and optimized every sequence labeling system to detect chemical entity mention by means the TPE. We showed that optimizing hyperparameter can be crucial. One sequence labeling system in our experiments gained an improvement of more than 65 %. We showed that a naive majority vote does not bring any improvement. For this reason, we introduced and evaluated `LSTMVoter`, a two-stage tool for combining underlying sequence modeling tools (as given by the NER of our comparative study). `LSTMVoter` achieved an improvement of up to 5 % compared to the best reference systems examined in our study. This two-level classifier appears to be capable of being further developed and improved by feeding it with the output of additional sequence labelling systems. In any event, our results and those of the other participants of BioCreative V.5 Task show that the task of NER of chemical entities has not been sufficiently solved yet. For a better recognition, a larger corpus should be generated so that today's popular deep learning algorithms can work on this data. A kind of human-in-the-loop architecture for automatic annotation and intellectual rework would also be helpful at this point in order to successively increase and improve the amount of data.

4.6. CRFVoter: Gene and Protein Related Object Recognition Using a Conglomerate of CRF-Based Tools

Hemati, Wahed and Alexander Mehler (2019a). "CRFVoter: gene and protein related object recognition using a conglomerate of CRF-based tools". In: *J. Cheminformatics* 11.1, 21:1–21:11. DOI: `10.1186/s13321-019-0343-x`. URL: `https://doi.org/10.1186/s13321-019-0343-x`.

4.6.1. abstract

Background Gene and protein related objects are an important class of entities in biomedical research, whose identification and extraction from scientific articles is attracting increasing interest. In this work, we describe an approach to the BioCreative V.5 challenge regarding the recognition and classification of gene and protein related objects. For this purpose, we transform the task as posed by BioCreative V.5 into a sequence labeling problem. We present a series of sequence labeling systems that we used and adapted in our experiments for solving this task. Our experiments show how to optimize the hyperparameters of the classifiers involved. To this end, we use various algorithms for hyperparameter optimization. Finally, we present `CRFVoter`, a two-stage application of CRF that integrates the optimized sequence labelers from our study into one ensemble classifier.

Results We analyze the impact of hyperparameter optimization of named entity recognition in biomedical research and show that this optimization results in a performance increase of up to 60 %. In our evaluation, our ensemble classifier based on multiple sequence labelers, called `CRFVoter`, outperforms each individual extractor's performance. For the blinded test set provided by the BioCreative organizers, `CRFVoter` achieves an F-score of 75 %, a recall of 71 % and a precision of 80 %. For the GPRO type 1 evaluation, `CRFVoter` achieves an F-Score of 73 %, a recall of 70 % and achieved the best precision (77 %) among all task participants.

Conclusion `CRFVoter` is effective when multiple sequence labeling systems are used and performs better than the individual systems collected by it.

4.6.2. Introduction

The research fields of biology, chemistry and biomedicine have attracted increasing interest due to their social and scientific importance and also because of the challenges arising from the intrinsic complexity of these domains. Like many other research areas, they are currently changing due to the rapid development of machine learning (ML) and artificial intelligence (AI). ML is used in many of these research areas. For instance, in the biomedical area it is used for biomedical signal processing (BSP) (Turner et al. 2017; Zhao and L. He 2014), biomedical imaging (BI) (Plis et al. 2014; Suk and Shen 2013; Qayyum et al. 2017) and disease prediction through patient profiling (Shickel et al. 2017). The former approaches work with structured data such as EEG data in the case of BSP. The last two approaches work with unstructured data such as MRI for BI and doctor-patient conversations in the case of disease classification and differential diagnosis (Mehler, Uslu, and Hemati 2016; Uslu, Miebach, et al. 2018; Reuber, Monzoni, et al. 2009; Reuber, Blackburn, et al. 2018). The growth in the amount of publicly available data has led to enormous efforts to develop, analyze and apply new learning methods in the field of chemistry and biology. This concerns, for example, virtual screening (Unterthiner et al. 2014) for drug design and drug discovery (Gawehn, Hiss, and Schneider 2016; Zhang et al. 2017). In order to advance areas of biological, chemical and biomedical research, it is important to perform state-of-the-art algorithms of data analysis. In carrying out scientific work, most researchers rely on published information to keep abreast of the latest developments in these fields, to avoid repetition and determine the direction of current studies. Numerous new publications appear daily in biomedical journals, in the form of scientific articles, patent applications, reports from health authorities and other text collections on the Internet, making it difficult to keep pace with the development of this discipline. Thus, there is an increasing interest in improving access to information on biological, chemical and biomedical data described in such texts and text repositories. To achieve this goal, a fundamental step is to automatically identify biological and chemical entities in these repositories. Based on this identification, interactions between drugs and proteins, for example, can be detected, side effects of chemical compounds and their associations to toxicological endpoints can be identified or information about metabolic reactions can be extracted (Emmert-Streib, Dehmer, and Haibe-Kains 2014).

For these reasons, initiatives and call for participation in corresponding competitions have been launched in recent years by professional communities that describe challenges in the identification of biochemical units. One of these initiatives is the BioCreative series which focuses on biomedical text mining. BioCreative is a “Challenge Evaluation”, where the participants are given defined text mining or information extraction tasks in the field of biology. These tasks include GM (Hirschman et al. 2005; Smith et al. 2008), GN (Hirschman et al. 2005; Morgan et al. 2008; Z. Lu et al. 2011), PPI (Krallinger, Vazquez, et al. 2011), CHEMDNER (Krallinger, Rabal, et al. 2015) and Chemical Disease Relation Extraction (CDRE) (Li et al. 2016; Wei et al. 2016) tasks.

The current *BioCreative V.5* task consists of two off-line tasks, namely *CEMP* and *GPRO*. CEMP requires the detection of chemical named entity mentions. The task requires detecting the start and end indices corresponding to chemical entities. The GPRO task requires identifying mentions of gene and protein related objects mentioned in patent titles and abstracts (Krallinger, Martin Pérez-Pérez, et al. 2017). In this work, we focus on the second task, that is, the GPRO task. The GPRO task is an abstraction of the well-known NER tasks, which can be reduced to a sequence labeling problem, where input sentences are represented as sequences of tokens. The task is then to tag genes and protein-related mentions in these sequences of sentences. The present paper addresses this task and is an extension of previous work (Hemati, Mehler, and Uslu 2017).

The paper is organized as follows: In Section 4.6.3 we describe our methodical apparatus and resources. First, we describe the data used for this work. We then present state-of-the-art tools for NER and how we adapted them for applying them in the biological domain. We examine the impact of hyperparameter optimization and show that it brings a considerable boost in performance. Next, we present a novel tool, called `CRFVoter`, for combining sequence labeling tools as used in our hyperparameter optimization. In Section 4.6.5, we present and discuss our results and in Section 4.6.6 we conclude and shed light on further work.

4.6.3. Methods

Dataset

The organizers of *BioCreative V.5* provided a corpus of 30 000 patent abstracts (titles and abstracts in English) from patents published between 2005 and 2014, where 21 000 of them are used as a training set and the remaining 9 000 as a test set. The corpus is manually annotated for the GPRO tasks. Gene and protein related object annotations were divided into type 1 and type 2. Type 1 are those GPRO mentions that can be normalized to database entries, like UniProt[17], NCBI[18], OMIM[19], GeneCards[20], FlyBase[21], etc.. Type 2 are those mentions that cannot be normalized. Table 4.7 shows the number of instances of type 1 and type 2 annotations in the GPRO Task. 5 795 documents from the

Table 4.7.: Number of instances of type 1 and type 2 in GPRO task.

Type 1	Number	Type 2	Number
ABBREVIATION	7 516	ABBREVIATION	27
FAMILY	1	FAMILY	5 029
FULL NAME	4 815	FULL NAME	27
IDENTIFIER	1	MULTIPLE	178
NESTED	89	NO CLASS	45
		SEQUENCE	23
Total count:	12 422	Total count:	5 329

[17] http://www.uniprot.org/

[18] https://www.ncbi.nlm.nih.gov/

[19] https://www.omim.org/

[20] https://www.genecards.org/

[21] http://flybase.org/

21 000 documents of the training set contained GPRO mentions. To reduce noise during training, only the annotated subset of 5 795 documents were considered; from now on, the collection of the documents will be called *filtered corpus*. Then, by means of random sampling, the filtered corpus was divided into three sets: 60 % of the document were sampled into the training set, 20 % into the development set and 20 % into the test set. The filtered corpus had been enriched with additional linguistic features. To this end, multiple preprocessing steps were applied on each of the three sets including sentence splitting, tokenization, lemmatization, part-of-speech tagging and fine-grained morphological tagging by means of the Stanford CoreNLP (Manning et al. 2014) and TextImager (Hemati, Uslu, and Mehler 2016). In addition, tokens were split on non-alphanumeric characters, as this variant brought a performance increase.

Table 4.8.: The number of documents, sentences and tokens of the filtered corpus.

# Documents	5 795
# Sentences	19 673
# Tokens	633 928

Table 4.8 lists the number of documents, sentences and tokens of the filtered corpus. Since the GPRO task can be reduced to a sequence labeling problem, the filtered corpus was converted into a sequence structure. To this end, a sequence of documents each containing a sequence of sentences each containing a sequence of tokens was constructed. This results in a file in TSV format, where each word and its associated features are in one line separated by tabs. Sentences are separated by an empty line. For the labeling of the GPRO mentions, the IOB tagging scheme (Ramshaw and Marcus 1995b) was used (I = inside of a entity, O = outside of a entity, B = beginning of a entity). This approach allows for the annotation of entities that span multiple tokens. Note that the beginning and end of each entity mention is marked. This allows models to not only learn tags themselves, but also the corresponding transition probability. Between all beginning and end tags, the inside parts, for example, should also be part of the manifestation of the entity. It is worth noticing that using the IOB scheme

has also disadvantages. The smallest unit that can be annotated is a token. Consider, for example, the token ”**B-Raf**V600E”. Only ”**B-Raf**” is annotated in the gold standard. This cannot be represented using the IOB format. To solve this problem, a tokenizer has to be developed that covers exactly these special cases. The filtered corpus contains 0,85% of these special cases. Since their recognition cannot be trained, they have been removed from the training set. However, during evaluation, these cases were considered as errors. In all experiments described in the following sections, we used the corpus as described so far.

4.6.4. System description

In this section we describe `CRFVoter`. Our approach implements a two-stage application of Conditional Random Fields (CRF) (Lafferty, McCallum, and Pereira 2001) using a conglomerate of sequence labelers for the detection of mentions of gene and protein related objects in biomedical patent abstracts. We trained and optimized five NER for tackling the GPRO task. We also optimized the hyperparameter settings of each of these NERs. Hyperparameter tuning is a challenging task in ML in the sense that the optimal set of hyperparameters depends on the model, the dataset and the domain (Claesen and Moor 2015) forming a huge interactive parameter space. In this context, our experiments focused on optimizing the hyperparameters of each NER system independently. This led to a noticeable increase of F-score compared to the default settings. For each NER, we performed a hyperparameter optimization by means of the *TPE* (Bergstra, Bardenet, et al. 2011). The NERs are more or less independent of each other in the sense that one can always find a subset of test cases being processed correctly by one NER but not by any other one. Therefore, combining these NERs is a promising candidate for increasing precision and recall. We started with computing combinations of these NERs by means of a simple majority vote (Dietterich 2000). Majority voting means to select the target label that is assigned by the majority of classifiers. Our experiments show that a simple majority vote brings no gain in precision and recall compared to the best performing reference systems being examined in our study. Thus, we alternatively experimented with a two-stage model, called `CRFVoter`, which trains a CRF to learn the best combination of the underlying sequence labeling tools (i.e. our case these are the NERs). We show, that `CRFVoter` outperforms every reference systems being examined in our study. In the rest of this section, we present a survey of hyperparameter optimization algorithms and discuss why TPE is the best optimization algorithm for our studies. We present a survey of NERs trained for the GPRO tasks and the parameter settings optimized by means of the TPE hyperparameter optimization algorithm. This includes the NER systems described in the following subsections. Finally we describe the ensemble classifiers based on majority voting and on our `CRFVoter`.

Hyperparameter Optimization

In this section, we describe the concepts of hyperparameter tuning. A ML model consists of various parameters that must be learned using the underlying training data. The main task of ML is to adapt a model to the given data. This process of fitting the model parameters to existing data is called *model training*. Hyperparameters are a class of parameters that cannot be learned directly from the training process. The hyperparameters are the variables that govern the training process itself. These parameters must be predefined; they define higher-level concepts about the model, such as complexity, convergence rate, penalty, and so on (Bergstra, Bardenet, et al. 2011). Hyperparameters are configuration variables of the training process that are normally kept constant. Hyperparameter optimization, also called hyperparameter tuning, is used to find optimal hyperparameter configurations for a ML algorithm on a given dataset. The goal is, to find optimized values for hyperparameters, which maximize the prediction accuracy of a model. Hyperparameter tuning works by performing several trials of the same training job. Each trial is a complete execution of the training process with values for pre-selected hyperparameters that are within predefined limits. Hyperparameter tuning optimizes one or more target variable where this variable is also called performance metric or hyperparameter metric (Hutter, Hoos, and

Leyton-Brown 2014). In our case we have considered a single target variable, that is, the F-score, because this is usually or at least predominantly done in NER. The hyperparameters are adjusted by running the entire training job, so that overall hyperparameter metric is improved. Since parameter spaces tend to include more and more dimensions, it is usually not possible to search the entire space to find the optimal configuration. Therefore, approximation algorithms must be used to maximize the hyperparameter metric (locally or globally). In the next sections we introduce a general notation and describe some hyperparameter optimization algorithms.

General notation Following the notation of Hutter, Hoos, and Leyton-Brown (2014) and Wistuba, Schilling, and Schmidt-Thieme (2015), a ML algorithm $\mathcal{A}$ is a mapping $\mathcal{A} : \mathcal{D} \rightarrow \mathcal{M}$ where $\mathcal{D}$ is the dataset and $\mathcal{M}$ is the space of all models. $\mathcal{A}$ has n hyperparameters, denoted as $\theta_1, \ldots, \theta_n$ and a configuration space $\Theta = \Theta_1 \times \ldots \times \Theta_n$ with $\theta_i \in \Theta_i, i = 1, \ldots, n$. The learning algorithm estimates a model $M(\boldsymbol{\theta}) \in \mathcal{M}$ that minimizes a loss function $\mathcal{L}$, given a hyperparameter configuration $\boldsymbol{\theta} = \langle \theta_1, ..., \theta_n \rangle$ on the training data $\mathcal{D}^{(train)}$:

$$\mathcal{A}_{\boldsymbol{\theta}}(\mathcal{D}^{(train)}) := \underset{M(\boldsymbol{\theta}) \in \mathcal{M}}{\arg\min} \mathcal{L}(M(\boldsymbol{\theta}), \mathcal{D}^{(train)}) \tag{4.1}$$

The goal of hyperparameter optimization is then to find the optimal configuration $\boldsymbol{\theta}^*$ using a validation set:

$$\boldsymbol{\theta}^* := \underset{\boldsymbol{\theta} \in \Theta}{\arg\min} \mathcal{L}(\mathcal{A}_{\boldsymbol{\theta}}(\mathcal{D}^{(train)}), \mathcal{D}^{(valid)}) \tag{4.2}$$

Grid Search Grid Search is a widely used hyperparameter optimization algorithm. It searches through a manually specified subset $\Theta_U \subset \Theta$ of the hyperparameter space. In a grid search, the set of trials is formed by assembling every possible configuration $\boldsymbol{\theta}$ of values in Θ_U, so the number of trials in a Grid Search is $|\Theta_U|$ elements (Bergstra and Bengio 2012). For each hyperparameter configuration $\boldsymbol{\theta} \in \Theta_U$ a model $M(\boldsymbol{\theta})$ is estimated and tested against the validation set $\mathcal{D}^{(valid)}$. This makes Grid Search suffering from the *curse of dimensionality*

(Bellman 2015) because the number of joint values in Θ_U grows exponentially with the number of hyperparameters. Since Grid Search works on a grid, continuous parameters must be discretized. In our experiments we used Grid Search in cases in which $|\Theta| < 200$ and where the parameter space did not contain continuous parameters – under these conditions, Grid Search will find the optimal configuration in foreseeable time.

Random Search Random Search is an optimization algorithm that searches a hyperparameter space Θ by selecting random hyperparameter configurations. Unlike Grid Search, no subset $\Theta_U \subset \Theta$ of the hyperparameter space must be defined. Instead, the parameters of a setting $\boldsymbol{\theta} \in \Theta$ are randomly selected. The advantage of this approach is that not only discrete parameters can be selected, but also continuous and mixed parameter spaces. Bergstra and Bengio (2012) found, that randomly chosen trials are more efficient for hyperparameter optimization then trials on a grid. They show empirically and theoretically that random searches are more effective for parameter optimization than grid searches when considering the same number of trials.

Bayesian Optimization Bayesian Optimization is a model-based optimization process for black box functions. The Bayesian optimization searches for the maximum of an unknown target function. It employs the Bayesian technique of setting a prior over the objective function and combining it with evidence to get a posterior function. Bayesian Optimization uses a Gaussian process (Rasmussen 2003) to model the surrogate. It optimizes the expected probability that new trials will improve compared to the best current observation. The Gaussian process is a distribution over functions, which involves adapting this distribution to the given data, so that functions are generated that come close to the observed data. This distribution is further optimized by iteratively selecting the next point, which must take into account both exploration (sampling from areas of high uncertainty) and exploitation (sampling areas likely to offer improvement over the current best observation) (Brochu, Cora, and Freitas 2010). Applied to hyperparameter optimization, Bayesian optimization builds a probabilistic model that assigns the hyperparameter values to the hyperparameter metric evaluated on the validation set. It has been shown that Bayesian optimization achieves better results in fewer trials than Grid Search and Random Search (Snoek, Larochelle, and Adams 2012).

Tree-structured Parzen Estimator The Tree-structured Parzen Estimator (Bergstra, Bardenet, et al. 2011) is a sequential model-based optimization (SMBO) (Hutter, Hoos, and Leyton-Brown 2011) approach. SMBO methods sequentially construct models to approximate the performance of hyperparameters based on "historical" (that is, preceding) measurements. For each iteration, TPE collects new observation, where at the end the algorithm decides which set of parameters it should try next. The main idea is similar to Bayesian Optimization (see Section 4.6.4). However, it fixes disadvantages of the Gaussian Process used by Bayesian Optimization. The TPE approach models $P(x|y)$ and $P(y)$ where x represents hyperparameters and y the associated hyperparameter metric. $P(x|y)$ is modeled by transforming the generative process of hyperparameters, replacing the distributions of the configuration prior with non-parametric densities. For the first few iterations TPE performs a Random Search. The next step is to divide the collected observations into two groups. The first group contains observations that yielded the best results after the evaluation and the second group contains the remaining observations. The goal is to find a set of parameters that are more likely to be in the first group and less likely to be in the second group. In contrast to Bayesian Optimization, TPE no longer relies on the best observation. Instead, a distribution over the best observations is used. The next step of the TPE is to model the likelihood probabilities for each of the two groups. This is the next big difference to the Gaussian Process. Gaussian Process models posterior probability instead of likelihood probability. Candidates are sampled using the likelihood probability from the group containing best observations. From the sampled candidates TPE tries to find a candidate that is more likely in the first group $l(x)$ and less likely in the second group $g(x)$; this is done by means of the *Expected Improvement* (EI):

$$EI(x) = \frac{l(x)}{g(x)} \tag{4.3}$$

From the sampled candidates, the parameter setting that has the highest Expected Improvement is selected for the next iteration. The optimization process ends after a predefined number of iterations.

Sequence labeling systems

In this section we describe the sequence labeling systems used in our experiments. These are state-of-the-art systems based on different architectures, namely CRF and Neural Networks. We show that hyperoptimization brings a considerable increase in performance. Finally, we present two variants for ensemble classifiers, namely Majority Voter and the `CRFVoter`.

Stanford Named Entity Recognizer Stanford Named Entity Recognizer[22] (StanfordNER) is a Java implementation of CRF based Named Entity Recognizer (Finkel, Grenager, and Manning 2005). Finkel, Dingare, et al. (2005) has participated in BioCreative to explore StanfordNER's limitations in the biological domain. They participated in BioCreative I Task 1A(Yeh et al. 2005) and achieved the best performance in the open task and the second best performance in the closed task. For StanfordNER our experiments are based on their results. The StanfordNER has since been further developed. New parameters have been added, which we have taken into account in our experiments.

[22] `http://nlp.stanford.edu/software/CRF-NER.shtml`

Table 4.9.: Parameter Space of Stanford Named Entity Recognizer used in our experiments. The column *Possible Values* describe the range of the parameters. The parameter setting with the best value is highlighted in bold.

Parameter	**Possible Values**
useClassFeature	[**true**,false]
useWord	[**true**,false]
useNGrams	[**true**,false]
noMidNGrams	[true,**false**]
normalizeTerms	[true,**false**]
usePosition	[**true**,false]
useNeighborNGrams	[true,**false**]
useMoreNeighborNGrams	[**true**,false]
usePrev	[true,**false**]
useNext	[**true**,false]
useTags	[**true**,false]
useWordPairs	[**true**,false]
useDisjunctive	[true,**false**]
useSequences	[**true**,false]
usePrevSequences	[**true**,false]
useNextSequences	[true,**false**]
useLongSequences	[**true**,false]
useTaggySequences	[**true**,false]
useSymWordPairs	[true,**false**]
useSymTags	[**true**,false]
useTypeSeqs	[**true**,false]
useTypeSeqs2	[**true**,false]
useTypeySequences	[**true**,false]
wordShape	**chris2useLC**
maxLeft	[1,**2**,3,4,5,6]
maxRight	[1,**2**,3,4,5,6]
maxNGramLeng	[1,2,3,**4**,5,6]
sloppyGazette	[true,**false**]
useGazFeatures	[**true**,false]
useWordTag	[**true**,false]
useWideDisjunctive	[**true**,false]
useLemmas	[**true**,false]
usePrevNextLemmas	[**true**,false]

Table 4.9 shows the corresponding hyperparameter space used in our experiments. Since the parameter space is so large that one cannot search it with a grid search, a hyperparameter optimization algorithm must be used. For our experiments we optimized the hyperparameters by means of TPE (see Section 4.6.4). During the optimization process we ran 200 trials to approximate the optimal parameter setting. The results of the trials are plotted in Figure 4.13 in the scatter plot. The scatter plot shows that the F-score converges towards 73 %.

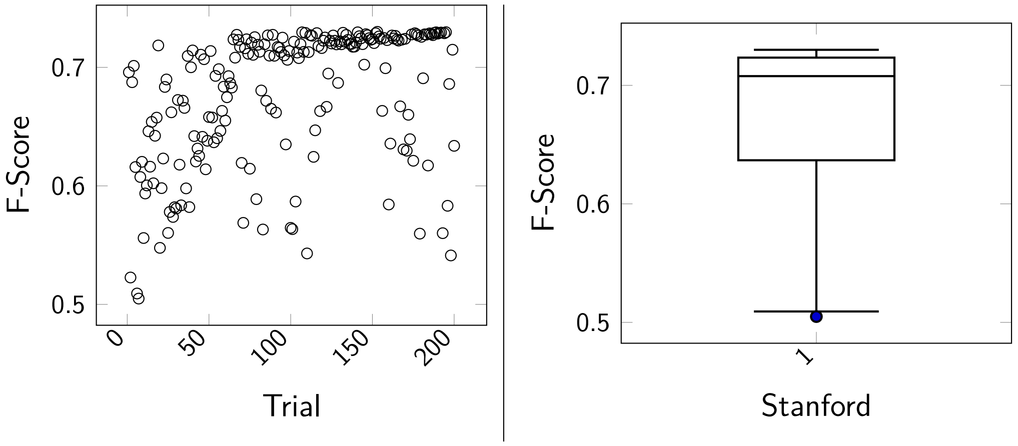


Figure 4.13.: The figure shows the results of optimizing StanfordNER by means of TPE. The scatter plot on the left side shows the results of each trial. The boxplot shows in which area the results are located and how they are distributed over this area. The difference between the best and the worst performing setting is 23%.

On the right side of Table 4.13 one sees the graphical representation of the F-Score distribution using a boxplot. The significance of a parameter study becomes immediately clear in this example. Depending on the parameter setting, the results vary by 23 %. The best performing set of features for GPRO, marked with bold font, leads to an F-score of 0,73. The worst setting results in an F-score of 0,50.

MarMoT MarMoT[23] is a generic CRF framework (T. Müller, Schmid, and Schütze 2013). It implements a higher order CRF with approximations such that

[23] http://cistern.cis.lmu.de/marmot/

it can deal with large output spaces. Additionally it can be trained to fire on the predictions of lexical resources (so-called gazette files) and on word embeddings (T. Müller, Schmid, and Schütze 2013; Mikolov, Sutskever, et al. 2013; Levy and Goldberg 2014; Ling et al. 2015; Komninos and Manandhar 2016). Table 4.10 shows the hyperparameter space used in our experiments for MarMoT. We ran 200 trials.

Table 4.10.: Parameter Space of MarMoT Tagger used in our experiments. The column *Possible Values* describe the range of the parameters. The parameter setting with the best value is highlighted in bold.

Parameter	**Possible Values**
Num iterations	[10,**20**]
Penalty	[**0**,1,2]
Beam size	[**1**,2,5]
Quadratic penalty	[**0**,1,2]
Order	[**1**,2,3,4]
Prob threshold	[0.01,**0.001**]
Effective order	[**1**,2,3]
Num chunks	[**2**,5,10]

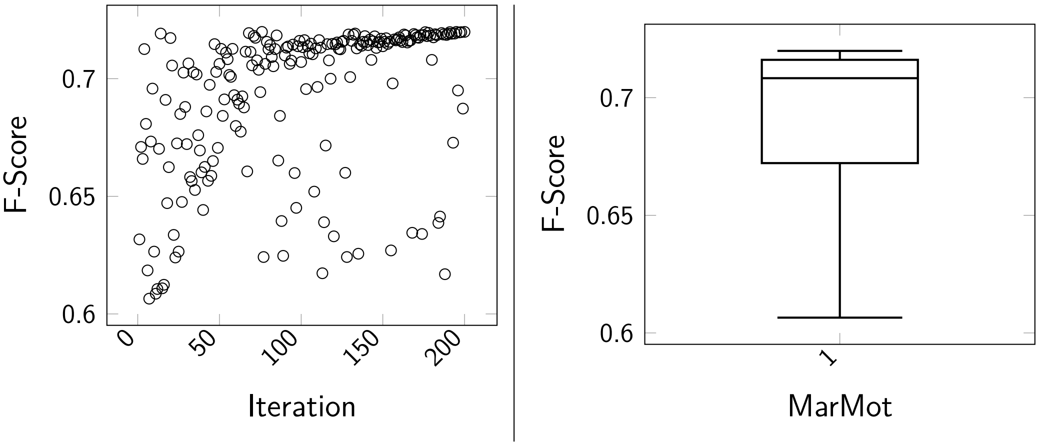


Figure 4.14.: The scatter plot on the left side of the figure shows the results of the optimization process of MarMoT. The boxplot shows in which area the results are located and how they are distributed over this area. Between the best and the worst setting are 11%.

The results of the iterations are shown in Figure 4.14 using a scatterplot. One can see that the F-score converges towards 0,72. The right side of Figure 4.14 shows the boxplot of the corresponding F-Score distribution. The best performing set of features for GPRO produces an F-score of 0,72. The worst set results in an F-score of 0,59. Once more, this difference hints at the importance of hyperparameter optimization.

CRF++ CRF++[24] is a customizable open source implementation of CRF (Kudo 2005). In our experiments with CRF++ we used unigram and bigram features including the current, the previous and the next word.

Table 4.11.: Parameter Space of CRF++ used in our experiments. The column *Possible Values* describe the range of the parameters. The parameter setting with the best value is highlighted in bold.

Parameter	**Possible Values**
c	[0.6, 1, 1.6, 3, 5, 7, **15**, 50, 100, 1000]
a	[CRF-L1, **CRF-L2**]

Table 4.11 shows the hyperparameter space used in our experiments for CRF++. The combination of parameters results in 20 model files, which is small enough to search the entire parameter space with Grid Search. The best performing set of parameters for GPRO generates an F-score of 0,69. The worst one results in an F-score of 0,04.

MITIE MITIE is an open source information extraction tool. MITIE can be trained using techniques like distributional word embeddings (Mikolov, Sutskever, et al. 2013; Levy and Goldberg 2014; Ling et al. 2015; Komninos and Manandhar 2016) and *Structural Support Vector Machines* (Geyer et al. 2016). Due to the lack of documentation, we did not optimize MITIE. The default configuration for named entity recognition produces an F-score of 0,65 for GPRO.

[24] `http://taku910.github.io/crfpp/`

Glample NER Tagger Glample NER Tagger is a neural-network-based named entity recognizer. It is based on Bidirectional LSTMs and CRFs (Lample et al. 2016). Due to the long-lasting training time, only the default parameter settings were considered. This resulted in an F-score of 0,74 for GPRO.

Majority Vote By means of majority voting, we combined the best performing outputs of each of the NER systems considered so far. We selected the label that was most frequently output by the different NER systems. Majority voting reaches an F-score of 0,68 for GPRO, which is below the best performing system considered so far. Facing these results we can state that a simple majority vote brings no gain in precision and recall. Therefore, we need an alternative considered next.

`CRFVoter` `CRFVoter` is a two-stage application of CRF using a conglomerate of sequence labelers. In the first step, each NER $c_m, m = 1..l$, is optimized independently on the training set, where the ith sequence t_i of length n of the set of training examples is of the form

$$t_i = \langle(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\rangle \tag{4.4}$$

$\vec{x}_j, j = 1..n$, is a feature vector corresponding to an element in the input sequence at position j – in our case this corresponds to a token. y_j is the corresponding discrete label of the element at position j – in our case this is the IOB2 formatted GPRO annotation label. The goal of a sequence labeling classifier c is to approximate the function $f(j) = y_j$ where y_j is the true label to be assigned to the input stream at position j. Approximations of f are computed by hyperoptimizing each classifier c as described above. After the training phase, a development set, which is independent of the training and the test set, is tagged by means of each NER c_m. The output label assigned by c_m is then taken by `CRFVoter` as an individual feature input. In the second step, `CRFVoter` combines each NER c_m into an ensemble classifier $c =$ `CRFVoter`$(\{c_1, c_2, \ldots, c_l\})$. The sequence of training examples used to train `CRFVoter` is of the form

$$t_i = \langle ((f_{c_1}(\vec{x}_1), f_{c_2}(\vec{x}_1), \ldots, f_{c_l}(\vec{x}_1)), y_1), \ldots, ((f_{c_1}(\vec{x}_n), f_{c_2}(\vec{x}_n), \ldots, f_{c_l}(x_n)), y_n\rangle \tag{4.5}$$

where $f_{c_m}(\vec{x}_j), m = 1..l, j = 1..n$, is the output label of classifier c_m computed for the input vector $\vec{x}_j$ at the jth position of the input sequence. That is, in stage one of `CRFVoter`, we calculate for each NER c_m and each token at position j of the input stream a corresponding output label $f_{c_m}(\vec{x}_j)$. In the second stage, these output labels are taken as features to feed our CRF operating on the same position j. In this way, we train `CRFVoter` based on a sequence of the latter feature sets, which is exemplified in Figure 4.15. Let x be the sequence of observed words in t_i and y be the sequence of states that correspond to the labels assigned in t_i. Linear-chain CRFs define the conditional probability of a state sequence to be (Lafferty, McCallum, and Pereira 2001):

$$P(y|x) = \frac{1}{Z_x} exp\left(\sum_{j=1}^{n}\sum_{m=1}^{l} \lambda_m f_m(y_{j-1}, y_j, x, j)\right) \tag{4.6}$$

Z_x is the normalization factor that makes the probability of all state sequences sum to one; $f_m(y_{j-1}, y_j, x, j)$ is a feature function, and λ_m is a learned weight associated with feature f_m. Feature functions measure the aspect of a state transition, $y_{j-1}, y_j \rightarrow yt$, and the entire observation sequence, x, centered at the current time step, j. Consider, for example, Figure 4.15. One feature function might have value 1 in cases where y_{j-1} denotes the state B-FULLNAME, y_j the state I-FULLNAME, and X_4 being the feature vector at position j. Large positive values for λ_m indicate a preference for such an event, whereas large negative values make the event unlikely. During tagging, `CRFVoter` takes again the output of each NER as input features and labels the sequence by means of the 2nd level CRF.

Our experiments show that `CRFVoter` brings 2% gain in F1-measure compared to the best performing reference systems being examined in our study. When operating on the blinded test set for GPRO provided by the BioCreative team, `CRFVoter` reaches an F-score of **0,75** for the evaluation of type 1 and of type

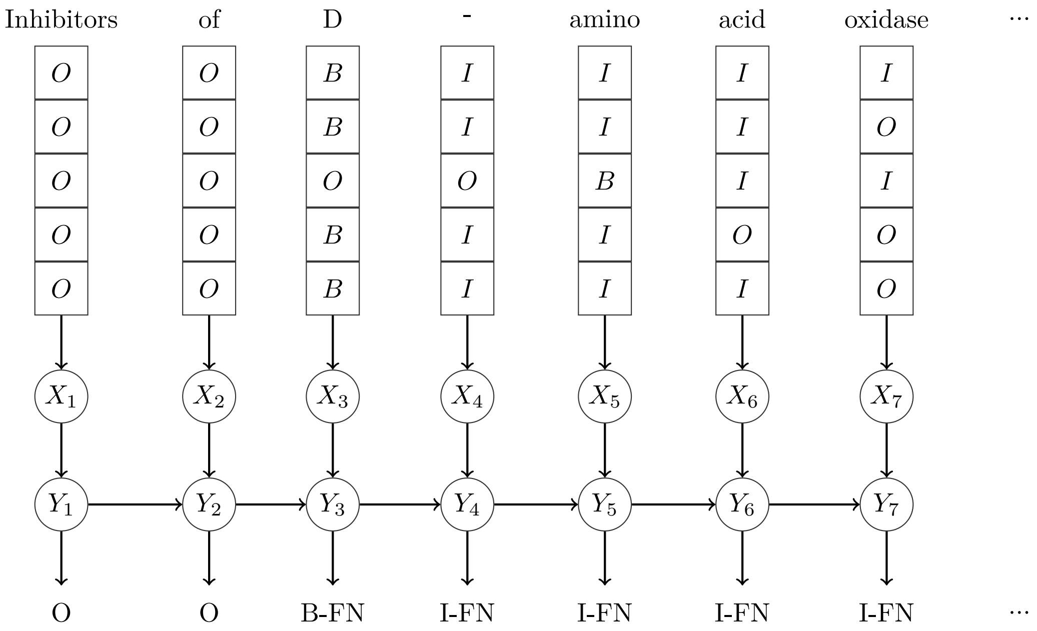


Figure 4.15.: Architecture of `CRFVoter` exemplified by means of a single sentence.

2.

4.6.5. Results

This section presents the results of our experiments for the GPRO task. For the evaluation of the GPRO Task the BioCreative Team has specified standard evaluation statistics, namely P, R and F (M Pérez-Pérez et al. 2017). Three main result types were examined. *False Negative*s (FN), that is, results corresponding to incorrect negative predictions. FN are cases that were part of the gold standard but overlooked by our systems. *False Positive*s (FP) are results of false positive predictions, that is, cases predicted by our system but not so marked in the gold standard. The third type of result is *True Positive*s (TP), i.e. results consisting of annotations predicted by our system and belonging to the gold standard as such. Recall is the fraction of correctly labeled positive results and all positive cases:

$$R = \frac{TP}{TP + FN} \tag{4.7}$$

Precision is the fraction of all correctly labeled positive results and all labeled results:

$$P = \frac{TP}{TP + FP} \tag{4.8}$$

F1-score is the harmonic mean of precision and recall:

$$F1 = 2 * \frac{P * R}{P + R} \tag{4.9}$$

In Section 4.6.4, the results of the hyperparameter optimization are visualized. For each sequence labeling tool, the hyperparameters were optimized using TPE or, if possible, using Grid Search. The results of the trials are plotted in scatterplots and the distribution of the results are visualized in the respective boxplots. The boxplots show the big spread of the outcomes of the trials during the respective optimization processes. For example, in the optimization process of CRF++, the difference between the worst to the best performer is 60%. The results show the need for ML algorithms to perform hyperparameter optimization.

Table 4.12.: Comparison of annotators trained an tested on the filtered corpus described in Section 4.6.3

System	**P**	**R**	**F**
Stanford NER	0,77	0,69	0,73
MarMoT	0,76	0,69	0,72
CRF++	0,75	0,64	0,69
MITIE	0,74	0,58	0,65
Glample	0,78	0,72	**0,74**
Majority Vote	0,64	0,72	0,68
`CRFVoter`	0,75	0,77	**0,76**

Table 4.12 shows the comparison of annotators trained for the GPRO task. The results listed are those obtained after the hyperparameter optimization described in Section 4.6.4, which were trained, optimized and tested on the corpus described in Section 4.6.3. Each sequence labeling system classifies a different subset correctly. Table 4.13 shows the pairwise differences between the sequence labeling

systems.

Table 4.13.: Differences of labeled output between each pair of NER system.

	Stanford	**MarMoT**	**CRF++**	**MITIE**	**Glample**
Stanford	0	2,29 %	2,12 %	2,44 %	2,50 %
MarMoT		0	2,56 %	2,61 %	2,43 %
CRF++			0	2,91 %	2,47 %
MITIE				0	2,51 %
Glample					0

The combination of the sequence labeling systems to a Majority Voter did not bring any performance increase and is even 5% below the best performer among the sequence labeling systems. In contrast, the `CRFVoter` increases the performance and is the best performer in our experiments. The performance values for the official BioCreative test set were created by training each model on the entire filtered corpus (see Section 4.6.3) and then evaluated on the official test set provided by BioCreative. For the blinded test set provided by the BioCreative organizers for GPRO, `CRFVoter` achieves an F-score of 75%, Recall of 71% and Precision of 80%. For the GPRO type 1 evaluation, `CRFVoter` achieves an F-Score of 73%, Recall of 70% and obtained the best precision (77%) achieved among all task participants.

Table 4.12 indicates that Glample and `CRFVoter` might be statistically tied. To investigate the significance of the improvements we used McNemars chi-square test (McNemar 1947) for *labeling disagreements* between Glample and `CRFVoter` with $\alpha = 0.05$. For both methods, we treated the predicted IOB-Tags for the test set that agreed with the gold annotations as positive, otherwise negative. For the McNemar test we only count the spans corresponding to biomedical named entities. We found that the comparison between Glample and `CRFVoter` is *significant* ($\rho < 0.05$) in terms of the test of (McNemar 1947).

4.6.6. Conclusion

In this work, we compared a set of sequence labeling systems. We trained and optimized every sequence labeling system for the GPRO task by means of several hyperparameter optimization algorithms and especially using the TPE. We showed that optimizing hyperparameter can be crucial. One sequence labeling system in our experiments gained an improvement of more then 60 %. We showed that a naive majority vote does not bring any improvement. For this reason, we introduced and evaluated the so-called `CRFVoter`, a two-stage CRF tool for combining underlying sequence modeling tools (as given by the NER of our comparative study). `CRFVoter` gained 2 % improvement compared to the best performing reference systems being examined in our study. Thus, `CRFVoter` may be further-developed by feeding it with the output of additional sequence labeling systems. A central theoretical outlook at this stage is to think about recursively organizing voters of the sort of `CRFVoter` beyond the first level by allowing different classifiers to contribute at different of these levels. In the past, such a procedure of recursive learning had been implemented by example of so-called semantic spaces (Rieger 1995) – see Gritzmann (2007) for such an approach. The theoretical background is to let the system systematically abstract the results of elementary learners: As with convolutional neuronal networks, this can help to find more and more abstract, but also increasingly characteristic representations of the input data. In any event, our results and those of the other participants of BioCreative V.5 Task show that the task of recognition of genes and protein-related objects has not yet been sufficiently solved. For better recognition, a larger corpus should be generated so that the nowadays popular Deep Learning algorithms can work on this data. A kind of human-in-the-loop architecture for automatic annotation and intellectual rework would also be helpful at this point in order to successively increase and improve the amount of data.

4.7. German Verb Sense Disambiguation

4.7.1. Introduction

Many words in a language are ambiguous and can therefore be interpreted differently. These interpretations are essentially dependent on the context in which they occur. Consider the following sentences:

- Das Gerät **läuft** einwandfrei. (*The device is operating correctly.*)
- Der Schaffner **läuft** zum Bahnhof. (*The conductor walks to the station.*)

The word *läuft* in the two sentences denotes different meanings: ”it works” and ”to walk”, respectively. While humans usually do not even think about the ambiguities of language, machines have to process unstructured text information and transform it into structured data to determine the underlying meaning. This computational identification of the meaning of a word is called Word Sense Disambiguation (WSD).

WSD is essential for many Natural Language Processing (NLP) applications. Without disambiguation, a word can only be considered as a sequence of letters irrespective of the meaning behind it. It has been shown that WSD can be helpful, if not essential, in various NLP tasks that require abstract semantic information. These include, for example, machine translation (Vickrey et al. 2005; Sudarikov et al. 2016; Neale et al. 2016), information retrieval (Zhong and H. T. Ng 2012; Chifu et al. 2015; H. T. Ng 2011) and question answering (Hung et al. 2005).

In addition, the disambiguation of verbs plays a special role. This is because verbs name events or states with participants and thus make them the organizational core of the sentence, so their meaning is the key to sentence meaning (Levin 1993).

Much research has been done on WSD, but mainly in English, including lexical resources (G. A. Miller 1995; Schuler 2006; Baker, Fillmore, and Lowe 1998) and sense annotated corpora (Edmonds and Cotton 2001; Snyder and M. Palmer 2004; Pradhan et al. 2007; Navigli, Jurgens, and Vannella 2013). Although the importance of WSD is indisputable, there is little research for German, especially for verbs. Henrich (2015) is the most comprehensive on the subject of Verb Sense Disambiguation (VSD). Various corpora were evaluated in the work, including manually annotated and automatically created ones. However, the focus was on all-word disambiguation. As a result, no high coverage for verbs was achieved.

The present work aims to fill this gap. We introduce a corpus which has a high coverage of annotated German verbs in terms of frequent verbs. Since the annotation of data is extremely time-consuming and therefore costly, various methods have been developed to semi-automatically generate quality data using translations, language models and Skinner's Law. The corpus contains hand annotated data for 985 ambiguous verb lemmas, which covers 80% of German verb tokens measured by verb occurrences in `COW` (Schäfer and Bildhauer 2012). We present neural network based tools for WSD and how they were adapted for VSD. In order to achieve comparable results, we reproduce Henrich (2015) experiments and compare our tools with theirs. In direct comparison to Henrich (2015), our best performing tool offers a performance increase of 6% and is therefore state-of-the-art.

The article is organized as follows: Section 4.7.2 describes our resources, materials and methodical apparatus. This includes the data and corpora used and created in our experiments. Section 4.7.3 presents state-of-the-art supervised tools for WSD and explain how they were adapted for the VSD task. Finally, we present our results, conclude and discuss future work in Section 4.7.4.

4.7.2. Materials and Methods

This section describes the manual sense annotation of selected corpora. The term *annotation* has different meanings within the digital humanities (Eggert, Lippert, and Etling 2019). In our project, the term is used for the process of marking segments of a text, verbs to be precise, as belonging to a category resulting in training data that can be used for machine learning. We also assume that such categories are pre-defined, adapted exploratively during the annotation process. Their recognition is based on the content of the text and on structure. This also means that the recognition of these categories is not trivial and requires text comprehension and some text interpretation. For the VSD Task this means that a list of possible meanings must be provided for each verb lemma to be disambiguated with as much information as possible about the meanings. Usually, these sense meanings are enumerated in so-called sense inventories. Several such sense inventories exist for German. Creating our own from scratch was out of the scope of this work, making the choice of the sense inventories used especially important.

Hence, the first step is to determine which sensory inventory is most appropriate for VSD in German. For this purpose, several German sense inventories were compared. The next step is to find out which corpus is most suitable. Different corpora were also compared. Since the annotation by human annotators is very resource intensive, different methods are introduced to minimize the annotation effort and to keep the quality of the data high at the same time.

Sense inventory

A sense of a word is a generally accepted meaning of a word (Kilgarriff 1997). A sense inventory divides these meanings of a word into its senses. However, word senses cannot simply be discretized, i.e. they are reduced to a finite discrete set of entries each encoding a distinct meaning. The main reason for this difficulty is that the language is affected by changes and interpretations. It is arguable where one sense ends and the other begins. Because of such uncertainties, different dictionaries give different detailed definitions of meaning. Therefore, the required granularity of the dictionary depends on the application. (Navigli 2009).

All traditional paper-based and machine-readable dictionaries are enumerated in a sense inventory. A description of all dictionaries and sense inventories used for German is out of the scope of this work. Here we give a brief overview of the most frequently used resources. Among them are the thesauri Duden (Duden, Berger, and Scholze 1980) and Wiktionary (*Wiktionary* n.d.) and the ontology GermaNet (Hamp and Feldweg 1997).

Duden is a spelling dictionary of the German language first published in 1 880. It is a classic paper-based dictionary. Lemmata are subdivided into senses. Duden senses are enumerated and further differentiated by an enumeration of finer grained word senses. The senses are combined with examples from German text corpora or with manually created examples. Verb entries also often contain lists of synonyms, with each list roughly corresponding to one sense of the verb. Unfortunately, the dictionary does not contain any relations on sense level, but only on the lemma level.

Wiktionary is a collaborative web-based dictionary under maintenance of wikimediafoundation. Word senses are enumerated and distinguished by descriptions and examples. For each sense, Wiktionary gives relationships such as synonyms, antonoms, hypernyms and hyponyms. Similar to Duden, the relations are not linked at sense level, but only at lemma level.

GermaNet is an ontology similar to the prominent English WordNet (G. A. Miller 1995). Word senses are grouped together into synsets which are connected by semantic relations. The subnet containing only verbs has a tree-like structure with hyponyms/hypernyms representing the child/parent relations.

The choice of an existing sense inventory is indispensable, both to keep the scope of the task manageable and to allow the annotated corpus to be used with existing tools or as an extension to existing corpora. The WordNet-like structure of GermaNet offers many advantages for machine processing, since relationships such as hyponymy, hyperonymy, antonymy and synonymy are represented. The thesauri also offer some of these relations, such as synonymy, but generally not hyperonymy or hyponymy. However, GermaNet is the only one that encodes these relations at the sense level. GermaNet is actively maintained, whereby several corpora with GermaNet annotations have already been annotated and are

Table 4.14.: The table shows statistics about the number of verb lemmas, synsets, senses and the COW coverage of Duden, Wiktionary and GermaNet. Note that Duden and Wiktionary do not code sensory-level relationships. Therefore, the senses are not combined into synsets, leaving the columns empty for the *# Verb Synsets* for Duden and Wiktionary.

	GermaNet	Duden	Wiktionary
# Verb lemmas	10 764	19 615	14 649
# Verb Synsets	14 178	∅	∅
# Senses	18 336	41 441	29 894
COW Coverage	99,1 %	99,4 %	98,5 %

already used in various tools (Henrich and E. W. Hinrichs 2013; Henrich, E. W. Hinrichs, and Vodolazova 2012; Henrich, E. W. Hinrichs, and Vodolazova 2011). New data with the same schema can be integrated into these existing tools.

Table 4.14 shows the number of lemmas and senses for the introduced sense inventories. It can be seen that the thesauri contain more lemmas. However, these verbs are only rarely used. This becomes apparent, since the 9500 verbs contained in the Duden, but not in GermaNet, only have a COW coverage of 0,32 %. The 6165 verbs contained in Wiktionary but not in GermaNet only have a COW coverage of 0,23 %.

The machine-readable form of GermaNet offers many advantages and has a sufficiently high COW coverage. Consequently GermaNet is used as sense inventory.

TTVCorp

One aim of this work is to create a reference corpus, from now on called TextTechnologyLab Verb Corpus (TTVCorp), in which a large number of verb occurrences are annotated. The aim is to annotate a relevant number of verb lemmas. At the same time, a sufficient number of example sentences per lemma should be annotated so that machine learning tools can work with them.

Since Henrich (2015) is similarly motivated, we base our work on it. As described in Section 4.7.2, GermaNet is used as sense inventory. The `TüBa-D/Z Treebank` is used as the underlying text resource for TTVCorp. `TüBa-D/Z` is a German newspaper corpus, which is processed semi-automatically with high-quality annotations on different linguistic levels (Telljohann et al. 2012). In addition, sentences from the `SALSA 2.0` Corpus (Burchardt et al. 2006) are added to TTVCorp, which are also provided with semantic annotations in Berkeley FrameNet format. The inclusion of `SALSA` has the advantage that predicate specific frames are provided. This makes it possible for future work to determine a correlation between used frames and verb senses.

The sense annotation in TTVCorp geared towards a high verb token coverage, rather then towards a high verb lemma coverage. This means that verbs were processed according to their rank-frequency distribution. The rank-frequency distribution was determined on the basis of the largest freely available German corpus, namely `COW` (Schäfer and Bildhauer 2012; Schäfer 2015). COW is an automatically processed web crawl corpus containing 807 782 354 sentences. Due to automatic processing, the COW corpus contains some lemmatization and pos-tagging errors. This explains the unusually high number of verb lemmas. To fix these errors, we applied the following heuristics:

- The verb lemma must be infinitive present tense, which means it has to end with ”-n”.
- The verb lemma must have at least 2 character
- The verb lemma must be lowercase

With these heuristics, 88 % of the verb lemmata were removed, while only 6 % of the verb tokens were removed (see Table 4.15).

On these purged data the ranking frequency distribution of the remaining verbs was determined. Figure 4.16 shows a cumulative rank-frequency distribution of verbs in `COW`. Zipf’s law (Newman 2005) applies here, which means that a relative small amount of verb lemmas covers a large amount of verb tokens. The 945 most common verb lemmas (modal and auxiliary verbs were excluded) cover 80 % of

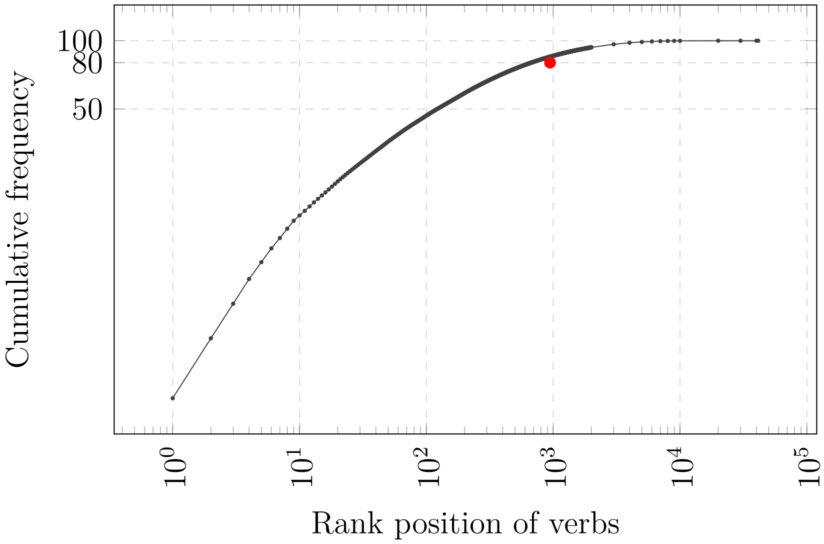


Figure 4.16.: The cumulative distribution of the token frequencies of the verbs in the COW corpus. The distribution reflects Zipf's law. The 945 most common verb lemmas cover 80 % of the verb tokens in COW.

the verb tokens. Due to restrictions on how much text can be annotated manually, we apply the Pareto principle (Newman 2005). Therefore we create an annotated corpus consisting of verb lemmas covering 80 % of the verb tokens in COW, that is 945 verb lemmas.

Table 4.15.: COW Statistics

	Plain	Filtered
# Verb lemmas	368 677	41 316
# Verb tokens	939 732 595	880 670 918
% of Hapaxe	50 %	35 %

Verbs have been selected in descending order of frequency according to the following criteria:

1. The selected lemmas have at least two senses in GermaNet.

2. The lemma is not already annotated in TüBa-D/Z.

3. The verb is not a modal verb and not an auxiliary verb.

Table 4.16.: Corpus statistics

	TüBa-D/Z	WebCAGe	deWaC	TTVCorp
# verb lemmas	79	959	15	945
# verb tokens	9 107	3 186	608	43 030
ave. frequency	115	3	41	46
ave. polysemy	2,8	3,7	7,9	3,54
COW coverage	6,2 %	66,4 %	6,4 %	76,4 %

Table 4.16 shows statistics on corpora that are already annotated with GermaNet senses, namely TüBa-D/Z (Telljohann et al. 2012), WebCAGe (Henrich, E. W. Hinrichs, and Vodolazova 2012) and deWaC (Raileanu et al. 2002). The table shows that none of the corpora offers a sufficiently high COW coverage, while at the same time a sufficient number of example sentences per verb lemma. To close this gap, TextTechnologyLab Verb Corpus (TTVCorp) was developed. TTVCorp offers a high COW coverage and has a sufficient number of sentences per verb lemma. In the following the annotation process of TTVCorp is presented and the quality of the corpus is evaluated. Since only a limited amount of annotated data can be generated, semi-automatic corpus expansion methods have been developed, which are presented below.

Annotation Process The annotation was performed with custom software (see Figure 4.19). Similar to (Kilgarriff 1998; C. Fellbaum et al. 2001; Saito et al. 2003; Passonneau et al. 2012; Henrich 2015), sentences were presented lemma-by-lemma. The sentences were preprocessed by TextImager to automatically capture additional information such as lemma, POS and dependency structures so that verbs with their corresponding senses from Germanet can be presented to the annotator. For each sense the hyponyms, hyperonyms, paraphrases and example sentences were listed, which was necessary to distinguish senses better from each other. Then sentences with the target verb from TTVCorp were presented to the annotator. For each target verb, one sense should be assigned where possible, or multiple senses can be assigned. Target verbs for which there is no appropriate sense in the sense inventory can be flagged accordingly. If multiple senses or no suitable sense are selected for a verb occurrence, this is an indication that the sense definition for this verb is problematic. In order to be able to evaluate the quality of the annotations, sentences were independently annotated by several annotators. A total of 19 annotators, including students, student assistants, doctoral students and postdocs from the field of text technology contributed to the construction of the annotated corpus.

One of the biggest challenges in natural language processing (NLP) is the shortage of training data. Since the annotation of data is extremely time-consuming and therefore costly, various methods have been developed to semi-automatically generate quality data using translations, language models and Skinner's Law.

Translation TTVCorp was enriched with machine-translated data. For this purpose, various semantically annotated English corpora (G. A. Miller 1995; Schuler 2006; Baker, Fillmore, and Lowe 1998) were translated into German. The English corpora use WordNet (Leacock and Chodorow 1998) as sense inventory. EuroWordNet (Vossen 1998) was used to map the WordNet Ids to GermaNet. EuroWordNet is a multilingual lexical database designed to connect WordNets of different languages, which made it possible to map English WordNet tags to GermaNet tags.

For translating the corpora from english to german fairseq (Ott et al. 2019) was used. Table 4.17 shows an example translation and the corresponding mapping of the target verb. Note that the translation is not forced to use the lemma of the converted GermaNet label. But the translated lemma should have a sense

Table 4.17.: Caption

English	German
The air conditioner was no longer running.	Die Klimaanlage lief nicht mehr.
WordNet: run%2:35:04::	GermaNet: 73 506

Table 4.18.: On the left are the initial sentences, where M is the masked word. On the right are expanded sentences, where the word in italics is BERT's prediction for the masked word in the original sentence.

Original	Expanded
Der Schaffner läuft zum [Bahnhof]$_M$.	Der Schaffner läuft zum *Bahnsteig.*
Der Schaffner läuft zum [Bahnhof]$_M$.	2. Der Schaffner läuft zum *Flughafen.*
Der Schaffner]$_M$ läuft zum Bahnhof.	3. *Er* läuft zum Bahnhof.
Das Gerät läuft [einwandfrei]$_M$.	1. Das Gerät läuft *automatisch.*
Das Gerät läuft [einwandfrei]$_M$	2. Das Gerät läuft *noch.*

closely synonymous to the converted label, if the translation is correct. This procedure added 31 585 sentences containing 64 178 labeled verbs to the TTVCorp. The conversion from WordNet to GermaNet labels is not one-to-one. Instances where multiple GermaNet labels were mapped to the same WordNet label are tagged with all possible GermaNet labels. On average, each instance has 2,36 labels.

Language Model State-of-the-art language representation models are using enormous amount of unannotated text from web-crawls for training general purpose models. These pre-trained models can then be used for downstream tasks such as question answering, sentiment analysis and text generation. For this work, text generation is relevant. With the help of the state-of-the-art language model, namely BERT (Devlin, M.-W. Chang, et al. 2018), the manually annotated data was expanded. For this, the straightforward technique of masking out some of the words in the annotated sentences was used and then condition each word bidirectionally to predict the masked words. BERT's bi-directional contextual model creates a representation of each masked word based on the other words in the sentence. Thus the masked words are replaced with semantically relevant words. Table 4.18 shows an example of the expansion process.

Skinner's law states that similar objects behave similarly. This behaviour has also been observed in human language (Mehler 2005). Neighbouring repeated words are used in the same way, ergo also have the same meaning. We have used this law to semi-automatically accelerate the annotation process. Algorithm 1 shows the process. For this the German Wikipedia was processed on different linguistic levels using the TextImager (Hemati, Uslu, and Mehler 2016). For each of the selected target verbs, Wikipedia articles W containing this target verb V were then determined. For each of the identified Wikipedia articles W, three sentences containing the target verb V were manually annotated. If all three target verbs were annotated with the same sense, it is assumed that all occurrences of the target verb in this article have the same sense. This sense is then automatically projected onto all other occurrences of the target verb. In order to evaluate the quality of the procedure, the senses of 10 verbs on 56 Wikipedia articles were manually assigned. The pair-wise agreement of the verb senses per article results in a score of 92 %. In Section A.2 the verb sense distribution per article can be found.

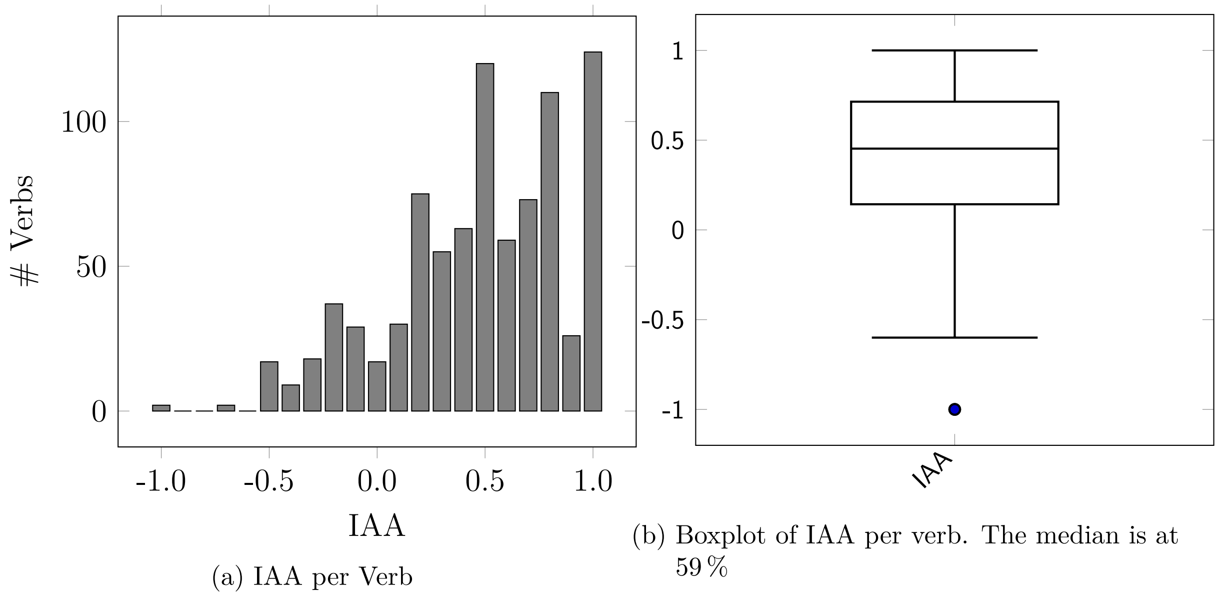
```
for each target verb = V do
    for each Wikipediaartikel = W containing V do
        #annotatedVerbs$_{VinW}$ = 0;
        trashhold = 3;
        senseIds = {};
        while #annotatedVerbs$_{vinW}$ < trashhold do
            S$_{vinW}$ = Sent. of W containing V;
            senseIds.add(annotate(S$_{vinW}$));
        end
        if len(senseIds) = 1 then
            for ∀ V in W do
                annotate V with senseIds[0];
            end
        end
    end
end
```

Algorithm 1: Algorithm for semi-automatic expansion of sense annotations using Skinner's Law applied to a Wikipedia corpus.

Results In this section the quality of the manually annotated data is evaluated. Very often it is not possible in linguistics to provide a definition with necessary and sufficient conditions for an annotation task. Even seemingly unambiguous annotation tasks such as POS tagging are not always unambiguous. As an annotator you make a decision for the annotation, but you can't say how certain you are about the annotation of this item. This is where the Inter-Annotator Agreement (IAA) comes in. With the IAA two things can be determined, namely reliability and validity of hand-coded data. The data is reliable if there is sufficient evidence that the annotators can agree on the categories assigned to the annotation units, insofar as this is determined by the purposes of the study (Krippendorff 2018). If different annotators consistently produce similar results, then it can be concluded that they have internalized a similar understanding of the annotation guidelines and that they work consistently under this understanding. (Krippendorff 2018; Artstein and Poesio 2008). Reliability is therefore a prerequisite for proving the validity of the annotation scheme, i.e. to show that the annotation scheme captures the "truth" of the phenomenon under investigation. That is, if the annotators are not consistent, some of them are either wrong or the annotation scheme is not appropriate for the data.

In the following we evaluate TTVCorp and GermaNet for reliability and validity. In the first step, we determine the IAA between each annotator and the `TüBa-D/Z` gold standard corpus.

To determine the reliability in TTVCorp, 30 % of the annotated data is annotated by at least two annotators. A proportional stratification is used to obtain a representative sample of data that needs to be annotated twice. This means that per verb lemma 30 % of the data is annotated by at least two annotators. The senses of the verbs are not clearly defined, annotators must construct criteria to distinguish the senses from the possible answers. This always requires a subjective ingredient that can be asserted as individually random. In the best case if the differentiation criterion proves itself it can then be applied consistently. An individually specific distribution of the possible answers is recognizable if the annotators either annotate a sufficient number of identical items or a representative sample from the same set of items. Such individual distributions should take



(a) IAA per Verb

(b) Boxplot of IAA per verb. The median is at 59 %

Figure 4.17.: Visualization of the IAA per verb before sense reduction.

into account *Randolph's Kappa* (Randolph 2005). As Artstein and Poesio (2008) mentions, WSD is one of the most difficult annotation tasks. Objective annotation tasks, such as POS tagging, syntax annotation and dialog act tagging, are called easy tasks. These tasks have a fixed number of classes that can be classified using decision rules and linguistic tests. These can be summarized in detail in an annotation guideline and provided with a sufficient number of examples per class (Brants 2000; Artstein and Poesio 2008). On the other hand, subjective tasks such as WSD and VSD are called hard tasks. The annotation decision is based on the intuition of the annotator. In the case of word sense tagging, different categories must be used for each word, making it impossible to write a single annotation manual with examples for all categories. The only possibility is to rely on a dictionary or sense inventory, which usually offers little information per sense. This lack of information makes it difficult for untrained annotators to make the fine-grained distinction between the senses, leading to uncertainty in annotation and a low IAA (Veronis 2001; Artstein and Poesio 2008). As Artstein and Poesio (2008) discusses, an IAA value of over 90 % is good for objective tasks and a value of over 70 % is significant for subjective tasks.

Figure 4.17 shows the distribution of the IAA per verb. The table shows that we have similar results to (M. Palmer, C. Fellbaum, et al. 2001) in terms of percentage and IAA score, namely 50 %. According to the scale of Artstein and Poesio (2008), this value is not sufficient. Kilgarriff (1999) recommends that only trained lexicographers should annotate in order to improve the IAA score. Since we do not have access to trained lexicographers, we cannot use Kilgarriff (1999)'s suggestion to improve the score. Instead, the proposal of M. Palmer, Dang, and C. Fellbaum (2007) was used. The approach is to address the problem of the inability of naive annotators to make fine-grained distinctions by introducing more coarse-grained sense definitions. Hard to distinguish verb senses were merged for this purpose to so-called super senses.

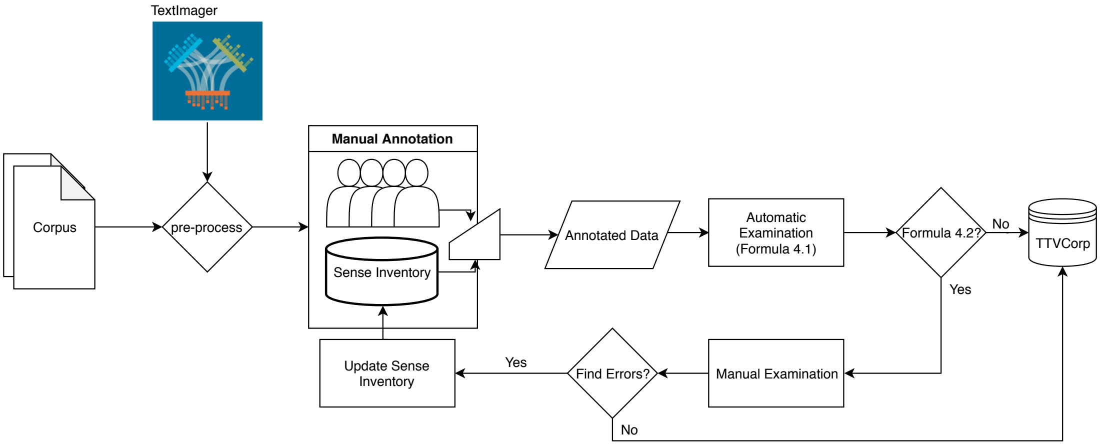


Figure 4.18.: Illustration of the annotation process for TTVCorp. The input corpus is preprocessed by TextImager and induced with POS, Lemma, Dependency information. Annotators are guided by a GUI in the annotation process. The annotations are automatically evaluated according to Formula 4.10 and Formula 4.10 in order to filter the merge candidates, which are then checked manually and, if necessary, the Sense Inventory is adjusted.

In order to identify which verb senses can be merged, we developed the procedure in Figure 4.18. The input corpus is processed by TextImager to enrich the data with additional information for the annotation process. The data is then manually annotated with the specified Sense Inventory. Verbs with IAA lower then 50 % were analyzed, as this is a strong indicator that the respective verb senses are difficult to distinguish. For this, coincidence matrices O were created according to the definition of Krippendorff (2018) for all annotated verbs.

For a lemma l with N multiply annotated sentences and possible senses V_l, O is a v-by-v, $v = |V_l|$ matrix that notes all pairs of annotation values, where each entry o_{ij} with $i, j \in V_l$ in such a matrix is given by:

$$o_{ij} = \sum_{u=1}^{N} \frac{\sum_{k \neq k'}^{m} I(v_{ku} = i) \cdot I(v_{k'u} = j)}{m_u - 1}, \tag{4.10}$$

where m_u is the total number of annotators that provided an answer for sentence u, v_{ku} is the answer provided by annotator k for sentence u and $I(\circ) = 1$ if $\circ$ is true, 0 otherwise. Senses were merged if they were respectively confused by the annotators. The identification of senses that are candidates for merging can be formally expressed as follows: For a given lemma l we compute the coincidence matrix O as above. Two senses i and j are candidates for merging if

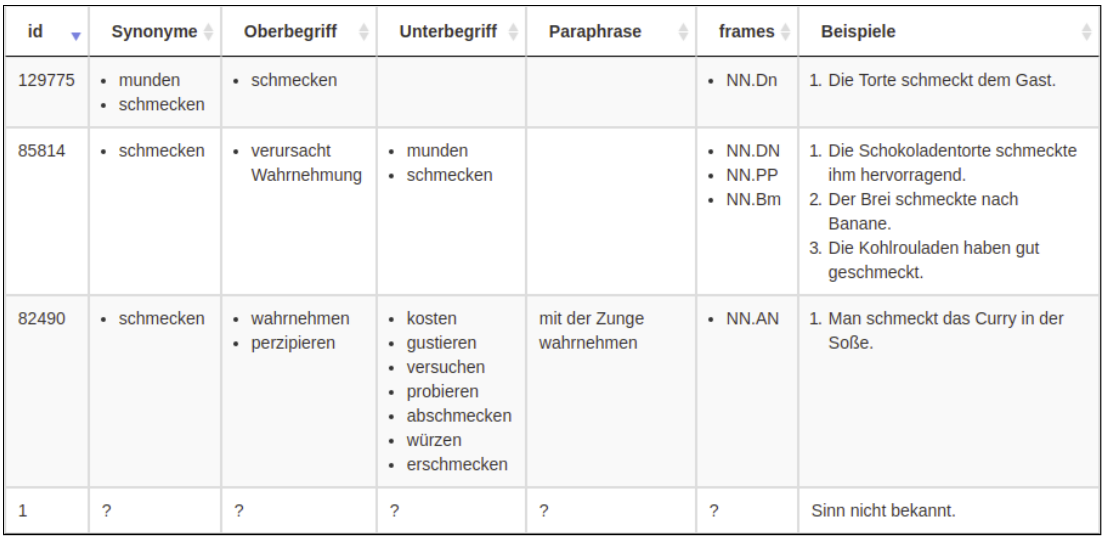
$$o_{ij} \geq \frac{\sum_j O_{ij}}{|V_l| + 1} \tag{4.11}$$

Note, that 1 is added to the number of senses to lower the threshold sufficiently to capture cases with only two senses. Otherwise, the senses of lemmata with only two senses would only be considered candidates for merging if the annotators agreed to exactly half of the respective cases. Note that this condition is not symmetric in i and j. Row i in O might indicate that i and j are candidates, but row j might not. In this case i and j are still candidates. In practice this often indicated hierarchical senses, such as *laufen* in the sense of movement on foot in general, *"to move"* vs. *laufen* in the sense of a fast, running movement, *"to run"*.

Following M. Palmer, Dang, and C. Fellbaum (2007)'s approach, the calculated candidates were reviewed by hand to merge the senses into super-senses. The following error classes and their resolution were identified during the manual merge process:

- Senses not distinguishable -> Merge Senses
- Duplicate/Circular Senses -> Merge Senses
- Senses/distinctions are missing -> Add missing sense
- Obsolete or dialectical meanings -> Mark as obsolete and remove sense
- Methaphor -> Mark as methaphor and remove from sense Inventory

The merge process is exemplified below using the example of *schmecken* (to taste). The initial senses, which are derived from GermaNet, are shown in Figure 4.19. The example sentence for the first sense is ”Die Torte schmeckt dem Gast. (*The cake tastes good to the guest*)”. The first sentence of the second sense is ”Die Schokoladentorte schmeckte ihm hervorragend. (*The chocolate cake tasted great to him*).” The example sentence for the final sense is ”Man schmeckt das Curry in der Soße. (*One could taste the curry in the sauce*).”. The distinction between the first two senses is very subtle and not apparent at all from the example sentences. Additionally, the top sense is a direct hyponym of the second sense. Tables 4.20 and 4.21 show the merging process. In this example, eight sentences were annotated by at least 2 annotators. The corresponding coincidence Matrix can be seen on the left side of Table 4.20. Note that multiple annotation is also possible ({85814∩129775}). Multiple annotations have been replaced by phantom annotators as shown in Table 4.19. The effect is that affected multiple anno-



id	Synonyme	Oberbegriff	Unterbegriff	Paraphrase	frames	Beispiele
129775	• munden • schmecken	• schmecken			• NN.Dn	1. Die Torte schmeckt dem Gast.
85814	• schmecken	• verursacht Wahrnehmung	• munden • schmecken		• NN.DN • NN.PP • NN.Bm	1. Die Schokoladentorte schmeckte ihm hervorragend. 2. Der Brei schmeckte nach Banane. 3. Die Kohlrouladen haben gut geschmeckt.
82490	• schmecken	• wahrnehmen • perzipieren	• kosten • gustieren • versuchen • probieren • abschmecken • würzen • erschmecken	mit der Zunge wahrnehmen	• NN.AN	1. Man schmeckt das Curry in der Soße.
1	?	?	?	?	?	Sinn nicht bekannt.

Figure 4.19.: Initial senses of ”schmecken” (to taste).

tations are explicitly weighted twice, because annotators have explicitly stated their indecision about these two senses.

Table 4.19.: Handling multiple annotations. Cases with three or more annotations are handled analogously. This causes the non-discrimination of these senses to be doubly weighted in the resulting Coincidence Matrix.

Annotator A
85814, 129775

⟶

Annotator A_1	Annotator A_2
85814	129775

The resulting coincidence matrix is shown on the right in Table 4.20. If the diagonal contains relatively low values, this is an indication that senses were annotated differently by annotators, thus not distinguishable, thus candidates for merging the sensens.

Table 4.20.: Coincidence matrices O for lemma $l =$ *schmecken*. On the left side, multiple annotations for a sentence u are allowed. On the right side these multiple annotations have been replaced by phantom annotators. This causes that affected multiple annotations are explicitly weighted twice.

	85814	82490	129775	{85814 129775}	Σ
85814	0	2	2	4	8
82490	2	4	0	4	10
129775	2	0	0	2	4
{85814 129775}	4	4	2	0	10
Σ	8	10	4	10	32

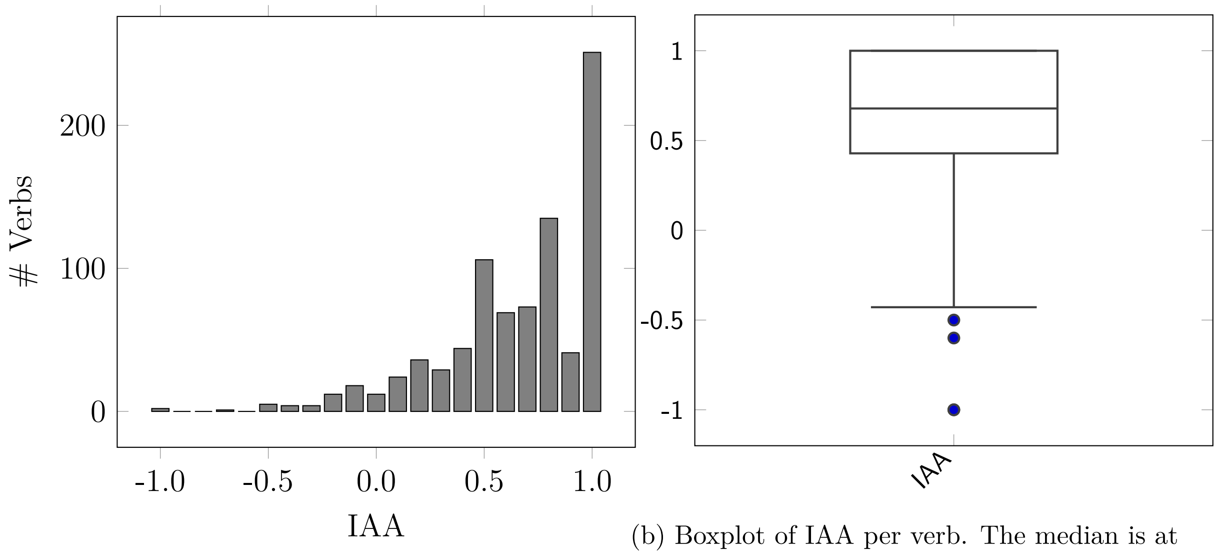
⟶

	85814	82490	129775	Σ
85814	4,667	4	9,333	18
82490	4	4	2	10
129775	9,333	2	2,667	14
Σ	18	10	14	42

Table 4.20 shows that the sense ids 85 814 and 129 775 were often confused, since $o_{85814,129775}$ and $o_{129775,85814}$ are relatively high in the coincidence matrix. This is an indication that senses are difficult to distinguish from each other. Expressed in Formula 4.11, it means the following: $o_{85814,129775} = 9.333 \geq \frac{\sum_j O_{85814,j}}{3+1} = 4.5$.

Table 4.21.: Final coincidence matrix after senses 85814 and 129775 were merged

	$S_{85814,129775}$	82490	$\sum$
$S_{85814,129775}$	26	6	32
82490	6	4	10
$\sum$	32	10	42



(a) IAA per Verb

(b) Boxplot of IAA per verb. The median is at 72 %

Figure 4.20.: Visualization of the IAA per verb after sense reduction.

That makes 85 814 and 129 775 candidates for merging. When reviewing the definitions of these senses, it also becomes apparent that precisely these senses have circular dependencies and are therefore difficult to distinguish. Table 4.21 shows the final result after the merging. This procedure was performed on all annotated verb lemmas to merge verb senses to coarse-grained super senses. The complete list of merged senses to supersenses can be found in Section A.2.

GermaNet is very mature and offers many advantages over other sense inventories. However, we have shown that some of the senses are too finely differentiated. With our presented method we can identify exactly these too fine-grained sense distinctions. After merging the senses into super senses, the IAA value is now 72 % (see Figure 4.20), which according to Artstein and Poesio (2008) is sufficient for semantic annotation tasks and can therefore be used for machine learning tools. In the following section state-of-the-art WSD tools are presented and how they have been adapted for VSD. In order to provide comparable results, the performance will be evaluated against Henrich (2015). Finally, the best performer will be trained and evaluated on TTVCorp. The outcome is a state-of-the-art supervised VSD with the largest coverage for German verbs.

4.7.3. Verb Sense Disambiguation

Verb Sense Disambiguation (VSD) is a sub-task of Word Sense Disambiguation (WSD). There are a variety of methods to solve the task of WSD. Following the notations of Henrich (2015), Navigli (2009), and Pal and Saha (2015) the methods can be grouped into knowledge-based approaches and machine-learning approaches.

Knowledge-based approaches make use of properties of lexical resources, including graph-based approaches (Agirre, Oier López de Lacalle, and Soroa 2018; Agirre, Oier Lopez de Lacalle, and Soroa 2014; Moro, Raganato, and Navigli 2014; Leacock and Chodorow 1998; Z. Wu and M. S. Palmer 1994; C. Fellbaum and G. Miller 1998) and information-content-based approaches (Lesk 1986; Resnik 1995; Jiang and Conrath 1997; Banerjee and Pedersen 2003; Chaplot and Salakhutdinov 2018).

Supervised approaches make use of sense-annotated training data, to learn how to predict the corresponding word senses for unseen data. The task is to build a machine-learned system using the human-labeled training data that can assign a dictionary sense from a predefined lexical resource (Henrich 2015; Papandrea, Raganato, and Bovi 2017; Luo et al. 2018; Peters et al. 2018a; Melamud, Goldberger, and Dagan 2016; Uslu, Mehler, Baumartz, et al. 2018).

Much research has been done on WSD, but mainly in English, including lexical resources (G. A. Miller 1995; Schuler 2006; Baker, Fillmore, and Lowe 1998) and sense annotated corpora (Edmonds and Cotton 2001; Snyder and M. Palmer 2004; Pradhan et al. 2007; Navigli, Jurgens, and Vannella 2013). WSD research for other languages, like German, is hampered by the lack of sufficient resources, in particular in the form of sense annotated corpus data. Henrich (2015) is one of the first that has concentrated extensively with WSD for German. It concentrates on WSD for German, uses GermaNet as a sense inventory and trains different supervised and knowledge-based systems on 3 different corpora. Knowledge-based systems have a greater coverage than supervised machine learning systems, whereas supervised learning systems outperform knowledge-based approaches to WSD by a big margin. The reason for greater coverage of knowledge-based systems is that they do not require annotated training data. In contrast, the supervised machine learning systems have achieved lower coverage because they have relied on annotated training data and are therefore only applicable to a limited set of lemmas.

This shortfall is closed by our work. Since the supervised WSD tools performed much better, we will concentrate on them in the following. State-of-the-art supervised WSD tools based on neural networks will be evaluated. The best performer from our study is then trained and evaluated on TTVCorp.

Supervised VSD

In this section, we compare different state-of-the-art supervised WSD systems that have been adapted to VSD. In order to produce comparable results, Henrich (2015)'s evaluation method was reproduced for the *TüBa-D/Z Gold Standard for Supervised WSD* Corpus. In contrast to Henrich (2015), our studies focus exclusively on verbs. Table 4.22 shows statistics of the corpus used.

Unlike Henrich (2015), we performed a 3-fold cross validation for our evaluation to minimize the chance that the results are an artifact of the split. It was ensured that training and test sets are representative samples of the data for each fold. This is achieved by proportionally stratifying all available annotations. That is, training and test portions for each fold are proportionally stratified, which means that the overall proportionality of class occurrences in the dataset is preserved (Henrich 2015; Botev and Ridder 2017; Witten, Frank, and Hall 2011).

Table 4.22.: TüBa-D/Z sense annotation subset for supervised WSD (Henrich 2015)

	Verbs
Total # of annotated word lemmas	68
Total # of tagged word tokens	8 540
Frequency range (occurrences/lemma)	24-799
Average frequency (occurrences/lemma)	126
Polysemy range in GermaNet (senses in GermaNet/lemma)	2-14
Average polysemy in GermaNet (senses in GermaNet/lemma)	2,9
Polysemy range of occurring words (occurring senses/lemma)	2-9
Average polysemy of occurring words (occurring senses/lemma)	2,6

Henrich (2015) has used traditional supervised classifiers, such as decision trees, probabilistic methods and SVMs. However, the current state-of-the-art for WSD is based on neural networks like FastSense (Uslu, Mehler, Baumartz, et al. 2018), Flair (Akbik, Blythe, and Vollgraf 2018) and Context2Vec (Melamud, Goldberger, and Dagan 2016). We adapted these models for the VSD task and evaluated them on the above mentioned corpus so that the results could be compared with those of (Henrich 2015). Table 4.23 shows the results of the evaluation. Note, that we outperformed Henrich (2015) and were able to set new state-of-the-art.

Table 4.23.: VSD results on TüBa-D/Z sense annotation subset for supervised WSD.

Classifier	F-Score
Context2Vec	76,04 %
Best of (Henrich 2015)	80,74 %
Flair	83,13 %
FastSense	**86,49** %

The best performer is used for the further experiments.

Supervised VSD on TTVCorp

This section explains how the best performer from the previous section, namely `FastSense`, was optimized for TTVCorp. A 3-fold cross-validation on TTVCorp was performed for the evaluation. Each fold was divided into training and test in such a way that the proportional stratification applies. In addition, the influence of the corpus expansion methods from Section 4.7.2 is evaluated.

`FastSense` consists of various parameters that must be learned using the underlying training data. The goal is to adapt a model to the given data from Section 4.7.2. This process of fitting the model parameters to existing data is called *model training*. Another class of parameters, the so called hyperparameters cannot be learned directly from the training process. The hyperparameters are the variables that govern the training process itself, which must be predefined and are configuration variables of the training process that are normally kept constant. They define higher-level concepts about the model, such as complexity, convergence rate, penalty, and so on (Bergstra and Bengio 2012). We perform hyperparameter optimization to find optimal hyperparameter configurations for `FastSense` on the TTVCorp, to find optimized values for hyperparameters, which maximize the prediction accuracy of our model. Finally, to optimize the hyperparameters, the TPE (Bergstra and Bengio 2012) implemented by `hyperopt` (Bergstra, Yamins, and Cox 2013) was used.

Table 4.24 shows the parameter space of the hyperparameter optimization.

Figure 4.21 show the results of each trial during the optimization process. The difference between the best and worst performer is 25%. This shows, that optimizing hyperparameter can be crucial.

After the optimization process, the influence of the corpus expansion was determined. For this purpose, the training data of the 3-folds cross-validation were enriched with the respective expansion methods of Section 4.7.2. Through our expansion methods, we have enriched the training corpus with semantically relevant information. This led to a significant increase in performance. It can be seen that the expansion has yielded an F-score increase of 3 %. For future semantic

Table 4.24.: Parameter Space of FastSense used in our experiments. The column *Possible Values* describe the range of the parameters. The parameter setting with the best value is highlighted in bold.

Parameter	**Possible Values**
epoch	[5,10,...,**40**,...,250]
wordNgram	[1,2,...,**10**]
minCount	[**1**,2,3]
learning rate	[0.1,...,**0.2**,...,1)]
loss	[**softmax**,hs,ns]
pretrainedVectors	[**true**,false]

Table 4.25.: Effect of corpus expansion on the performance of tagger.

Corpus	F-Score
TTVCorp	75 %
TTVCorp + Skinner + Translation + LM	78 %

NLP tasks it is therefore indispensable to use these expansion methods.

4.7.4. Conclusion and Future Work

In this work, we present the largest sense annotated corpus for German verbs. The selection of the annotated data was determined using Zipf's law and the Paredo principle. The subset of verb lemmas that covers 80% of verb tokens was annotated. At the current state, these are 985 ambiguous verbs, with a total of 42 944 annotated sentences. The reliability and validity the data is evaluated using Randolph's Kappa. In the process, it turned out that some verbs have low IAA score. The reason for this is that the distinction of senses in GermaNet is too fine-grained for these verbs. In order to solve this problem, we have developed a method to locate and merge these senses, which are difficult to distinguish from each other. This resulted in an IAA increase from 59 % to 72 %, which according to (Artstein and Poesio 2008) is sufficient for semantic annotation tasks.

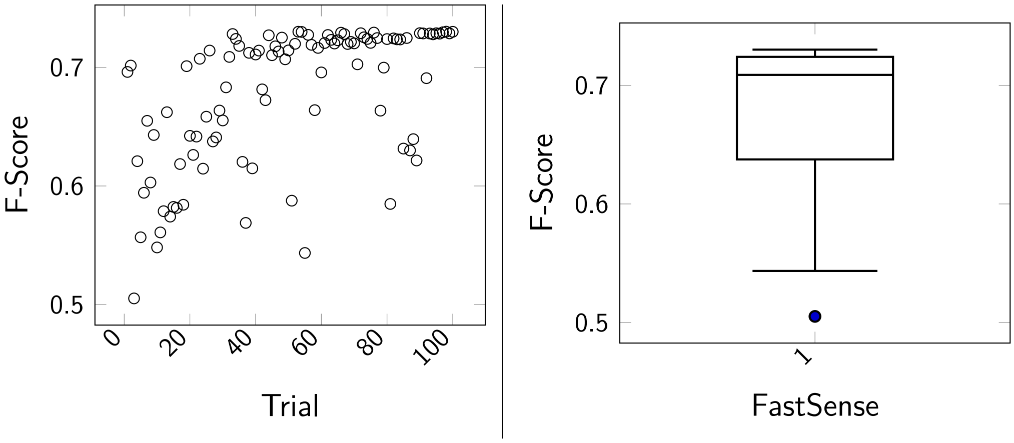


Figure 4.21.: The figure shows the results of optimizing FastSense on TTVCorp by means of TPE. The scatter plot on the left side shows the results of each trial. The boxplot shows in which area the results are located and how they are distributed over this area. The difference between the best and the worst performing setting is 23%.

Various neural network based taggers were trained and evaluated. Compared to (Henrich 2015), our Best Performer has achieved a performance increase of 6% F-Score (80% to 86%) and is therefore state-of-the-art. The evaluation on TTVCorp gives an F-Score of 73%.

Since annotation is very time-consuming and expensive, automatic and semi-automatic methods for corpus expansion have been developed. These include expansion through Skinners Law, Translation and Language Models. The addition of the expanded data to the train set resulted in a performance increase from 73 % to 76 %. Many classification tasks suffer from insufficient training data. For these classification tasks, especially for semantic classification tasks, it may be useful to use our proposed expansion methods.

In future work we will increase the quantity and quality of TTVCorp.

To reproduce the experiments, we put the Tagger, the models, the corpora and all the evaluations online on GitHub. An online demo of the tagger can be accessed at the following address:

`https://textimager.hucompute.org/VerbsAnnotator/input.html`

5. Concluding Remarks

The overall aim of this thesis was to develop a framework for the NLP landscape enabling different specialised NLP tools to become interoperable, namely TextImager. In order to be able to process the ever-increasing amounts of data, the framework is able to run cluster-based on several machines whilst being scalable both horizontally and vertically. Thereby specialized tools for the individual tasks in the process sequence are developed by different organizations. TextImager merges existing systems and development environments in a modular fashion to exchange the work packages with each other in order to exploit specialisation advantages. With TextImager it is now possible to solve the big and diverse unstructured data problem. This is solved using the 6 paradigms Multi-Service, Multi-Server, Self-Orchestration, Multi-Database, Authority Management, Multi-Representation. Table A.1 shows the processing time with the TextImager for the corpus presented in Section 3.1. It should be noted that the TextImager's architecture has made processing 7-fold faster. This factor can be further increased as the infrastructure now allows to dynamically boot additional instances.

TextImager is already used by many works as a pre-processing pipeline, as a feature generator and as a programming interface (Mehler, Abrami, et al. 2018; Baumartz, Uslu, and Mehler 2018; Hemati and Mehler 2019a; Hemati and Mehler 2019b; Kett et al. 2018; Abrami, Mehler, Lücking, et al. 2019; Abrami, Mehler, and Spiekermann 2019; Rutherford, Hemati, and Mehler 2018; Hunziker et al. 2019; Uslu, Mehler, and Baumartz 2019; Uslu and Mehler 2018; Uslu, Mehler, and Meyer 2018; Mehler, Hemati, Uslu, et al. 2018; Uslu, Hemati, et al. 2017; Hemati, Uslu, and Mehler 2017). The project is freely accessible on GitHub and is being further developed by many contributors already. One of the future works

Table 5.1.: Statistics about the time and space complexity of the processed corpus introduced in Section 3.1 processed by TextImager. Compared to the non-distributed version, Textimager has become more than 7 times faster.

	Step	Days	GB
1	Tokenize	0.32	82.48
2	Lemmatization	0.84	97.70
3	POS Tagging	0.37	94.42
4	Named Entity Recognition	0.47	9.06
5	Dependency Parsing	1.85	225.46
6	Time Recognition	0.99	11.95
7	Sentiment Analysis	0.76	7.13
8	Semantic Role Labeling	0.9	8.09
9	Wikification	1.24	26.40
10	Coreference	2.58	6.74
	$\sum$	10.32	569.43

will be to develop a process for automating the integration of new tools[1] so that TextImager is always at the cutting edge of research.

In the course of the development of computational linguistics, a rudimentary process model has established itself as a scientific discipline that defines practiced sequence regularities of NLP tools. The advantage of such a NLP pipeline is that more complex downstream tasks can be subdivided into smaller tasks, for which specialized tools can be developed. However, it also has its disadvantages. Errors in the processing sequence accumulate. To avoid this, the entire processing would have to be formulated as a single optimization/learning problem (Mehler, Hemati, Gleim, et al. 2018). Individual steps can thus influence each other better. For example, semantic analysis shows that a certain part of speech is implausible for a term. There are already approaches that carry out this type of single optimization, namely multi-task learning (MLT). Inline with (Sener and Koltun 2018) a MLT problem can be spanned over an input space X and a collection of task spaces $\{Y^t\}_{t \in |T|}$, such that a data set of independent and identically distributed points $\{x_i, y_i^1, ..., y_i^T\}_{i \in |N|}$ is given where T is the number of tasks, N

[1] https://paperswithcode.com/area/natural-language-processing

is the number of data points and y_i^T is the label of the t^{th} task for the i^{th} data point. Further consider a mapping function per task as $f^t(x; \theta^{sh}, \theta^t) : X- > Y^t$, such that shared parameters between tasks (θ^{sh}) and task-specific parameters (θ^t) exists. The task specific loss function is defined as $\mathcal{L}^t(.,.) : Y^t \times Y^t \rightarrow \mathbb{R}^+$ The loss-function of the MLT can be defined as:

$$\min_{\theta^{sh}, \theta^1, \ldots, \theta^T} \sum_{t=1}^{T} c^t \hat{\mathcal{L}}^T(\theta^s h, \theta^t) \tag{5.1}$$

where c^t are the computed weights per task, and $\hat{\mathcal{L}}^T(\theta^{sh}, \theta^t)$ the empirical loss of task t. The problem with MLT is, that it is not possible to define a global optimal setting. Consider two sets θ and $\bar{\theta}$, where solution θ is better for task t_1 and $\bar{\theta}$ is better for task t_2. Typically there is no pairwise importance of tasks available, which makes it impossible to compare two solutions. To counteract this, a MLT can be formulated as multi-objective optimization, where the goal is achieving Pareto optimality:

(a) solution θ is no worse then $\bar{\theta}$ in all tasks

(b) solution θ is better then $\bar{\theta}$ in at least one task

Interest in multi-task learning in NLP is growing, as reflected by the publication of more and more shared tasks and test suites for MLT. Among them are DecaNLP (McCann et al. 2018) and GLUE (Wang et al. 2018). Transformer models such as `BERT` (Devlin, M. Chang, et al. 2019), `XLNet` (Zhilin Yang et al. 2019) and `ELMo` (Peters et al. 2018b) are the current best performers in the above mentioned tasks. Due to their architecture coupled with the huge amounts of data, their representation possibilities become very powerful. However, this is also one of the big disadvantages. The information is implicitly stored in the heads of the individual layers, which cannot be interpreted explicitly.

As future work the two worlds must be combined with each other. The explicit pipeline and task oriented sequential NLP processing, as currently implemented by TextImager, must be combined with transformer models in a dynamic multi-task learning process described in Formula 5.1. A possible solution would be a

Hierarchical Model for Multi-task learning (Sanh, Wolf, and Ruder 2019). The model is hierarchically trained to introduce an inductive bias by monitoring a series of low-level tasks (i.e. lemmatization and POS-tagging) in the lower layers of the model and more complex tasks (i.e. semantic role labeling, entity mention recognition, etc.) in the upper layers of the model. The goal of such a model would be to minimize the lossfunction introduced in Formula 5.1. In contrast to the black box transformer models, the parameter setting θ^t could be determined for each task T explicitly, thus can be better interpreted.

A big step into the semantic analysis of verbs for German was done in this work. A state-of-the-art verb sense disambiguation framework for German has been developed. To accomplish this, the largest sense-annotated German verb corpus was built, which is needed to train supervised neural network VSDs. The corpus covers 80% of the German verb tokens found in COW. However, some remaining verbs from GermaNet are not yet covered. In addition, there are verbs that are not listed in GermaNet. Therefore, it is not sufficient to operate on the basis of manually annotated data in order to be able to cover verbs senses in their entirety, as the creation is too costly. In order to get this under control, more work must be done with distributional and declarative semantics in the context of VSD. Distributional semantics is a field of research that develops and investigates theories and methods for quantifying and categorizing semantic similarities between linguistic elements, based on their distribution properties in large corpora. It can be summarized in the distribution hypothesis: linguistic elements with similar distributions have similar meanings (Harris 1954; Dahl 2016). Distributional models represent a word through the contexts in which they occur, whereby modern approaches implement vector space models or word embeddings (Mikolov, Chen, et al. 2013; Blei, A. Y. Ng, and Jordan 2003; Deerwester et al. 1990). In these models words are represented as points in a high-dimensional space, in which similar words tend to be closer. The downside is, that one vector is created per syntactic word, which means that the sense level is not captured. Current models, such as `BERT` (Devlin, M. Chang, et al. 2019), `XLNet` (Zhilin Yang et al. 2019) and `ELMo`, induce context-sensitive word embeddings. The idea is to create sense embeddings implicitly, which are represented by different vectors for the same syntactic word in different contexts. In future work we will use this kind of implicit semantics to obtain sense distinctions for verbs for which no training data is available and therefore no supervised ML can be trained.

Another way to accomplish a higher coverage of verb semantics is to use declarative semantics. In the case of German verbs, particle verbs are particularly useful because they are highly productive (Springorum, Utt, and Schulte im Walde 2013). The greatest challenge of these verbs is that of compositionality: can the meaning of such a verb be predicted on the basis of general principles from that of the particle or prefix and the rest? Hence, it remains an open question, whether an algebraic model can be developed to approximate the meaning, based on the combination of the meaning of the base verb and the particle (Bott and Schulte im Walde 2018; Köper et al. 2016). However, there has been no research on compositionality and context-sensitive subwordembeddings, that will be subject of focus in future work.

Subcategorization and verb senses are known to be linked. Levin (1993) widely used verb classification is based on the hypothesis that the syntactic behavior of a verb and its meaning are strongly linked. Similar to Levin (1993) classes, German verbs can be subdivided into semantic verb classes in order to assign unknown verbs and non-literal verbs semantically (Schulte im Walde 2006; Scheible et al. 2013). In future work we will combine the novel contextualized embeddings in combination with subcategorization frames and predicate-argument-structure of verbs (W. Wagner, Schmid, and S Schulte Im Walde 2009; Schulte im Walde et al. 2010) to extract selectional restrictions (Weller, Sabine Schulte Im Walde, and Fraser 2014; Mu, Hartshorne, and O'Donnell 2017) for cluster-based sense disambiguation and induction.

Bibliography

Abrami, Giuseppe, Alexander Mehler, Andy Lücking, Elias Rieb, and Philipp Helfrich (2019). “TextAnnotator: A flexible framework for semantic annotations”. In: *Proceedings of the Fifteenth Joint ACL - ISO Workshop on Interoperable Semantic Annotation, (ISA-15).* ISA-15. Gothenburg, Sweden.

Abrami, Giuseppe, Alexander Mehler, and Christian Spiekermann (2019). “Graph-Based Format for Modeling Multimodal Annotations in Virtual Reality by Means of VAnnotatoR”. In: *HCI International 2019 - Late Breaking Posters - 21st HCI International Conference, HCII 2019, Orlando, FL, USA, July 26-31, 2019, Proceedings*, pp. 351–358. DOI: `10.1007/978-3-030-30712-7\_44`. URL: `https://doi.org/10.1007/978-3-030-30712-7%5C_44`.

Aggarwal, Charu C. and ChengXiang Zhai, eds. (2012). *Mining Text Data.* Springer. ISBN: 978-1-4419-8462-3.

Agirre, Eneko, Oier Lopez de Lacalle, and Aitor Soroa (2014). “Random Walks for Knowledge-Based Word Sense Disambiguation”. In: *Computational Linguistics* 40.1, pp. 57–84. DOI: `10.1162/COLI\_a\_00164`. URL: `https://doi.org/10.1162/COLI%5C_a%5C_00164`.

Agirre, Eneko, Oier López de Lacalle, and Aitor Soroa (2018). “The risk of sub-optimal use of Open Source NLP Software: UKB is inadvertently state-of-the-art in knowledge-based WSD”. In: *CoRR* abs/1805.04277. arXiv: `1805.04277`. URL: `http://arxiv.org/abs/1805.04277`.

Akbik, Alan, Duncan Blythe, and Roland Vollgraf (2018). “Contextual String Embeddings for Sequence Labeling”. In: *COLING 2018, 27th International Conference on Computational Linguistics*, pp. 1638–1649.

Artstein, Ron and Massimo Poesio (2008). “Inter-Coder Agreement for Computational Linguistics”. In: *Computational Linguistics* 34.4, pp. 555–596. DOI:

10.1162/coli.07-034-R2. URL: https://doi.org/10.1162/coli.07-034-R2.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* URL: http://arxiv.org/abs/1409.0473.

Baker, Collin F., Charles J. Fillmore, and John B. Lowe (1998). “The Berkeley FrameNet Project”. In: *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, COLING-ACL ’98, August 10-14, 1998, Université de Montréal, Montréal, Quebec, Canada. Proceedings of the Conference.* Pp. 86–90. URL: http://aclweb.org/anthology/P/P98/P98-1013.pdf.

Banerjee, Satanjeev and Ted Pedersen (2003). “Extended Gloss Overlaps as a Measure of Semantic Relatedness”. In: *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pp. 805–810. URL: http://ijcai.org/Proceedings/03/Papers/116.pdf.

Baumartz, Daniel, Tolga Uslu, and Alexander Mehler (2018). “LTV: Labeled Topic Vector”. In: *COLING 2018, The 27th International Conference on Computational Linguistics: System Demonstrations, Santa Fe, New Mexico, August 20-26, 2018*, pp. 142–145. URL: https://www.aclweb.org/anthology/C18-2031/.

Bellman, Richard (2015). *Adaptive Control Processes - A Guided Tour (Reprint from 1961).* Vol. 2045. Princeton Legacy Library. Princeton University Press. ISBN: 978-1-4008-7466-8. DOI: 10.1515/9781400874668. URL: https://doi.org/10.1515/9781400874668.

Bergstra, James, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl (2011). “Algorithms for Hyper-Parameter Optimization”. In: *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.* Pp. 2546–2554. URL: http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.

Bergstra, James and Yoshua Bengio (2012). “Random Search for Hyper-Parameter Optimization”. In: *J. Mach. Learn. Res.* 13, pp. 281–305. URL: http://dl.acm.org/citation.cfm?id=2188395.

Bergstra, James, Daniel Yamins, and David D. Cox (2013). “Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures”. In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pp. 115–123. URL: http://proceedings.mlr.press/v28/bergstra13.html.

Biemann, Chris and Alexander Mehler, eds. (2014). *Text Mining - From Ontology Learning to Automated Text Processing Applications.* Theory and Applications of Natural Language Processing. Springer. ISBN: 978-3-319-12654-8. DOI: 10.1007/978-3-319-12655-5. URL: https://doi.org/10.1007/978-3-319-12655-5.

Bird, Steven, Ewan Klein, and Edward Loper (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* Beijing: O’Reilly. ISBN: 978-0-596-51649-9. DOI: http://my.safaribooksonline.com/9780596516499. URL: http://www.nltk.org/book.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). “Latent Dirichlet Allocation”. In: *J. Mach. Learn. Res.* 3, pp. 993–1022. URL: http://jmlr.org/papers/v3/blei03a.html.

Botev, Zdravko and Ad Ridder (2017). “Variance Reduction”. In: *Wiley StatsRef: Statistics Reference Online.* American Cancer Society, pp. 1–6. ISBN: 9781118445112. DOI: 10.1002/9781118445112.stat07975. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118445112.stat07975. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118445112.stat07975.

Bott, Stefan and Sabine Schulte im Walde (2018). “German particle verbs: Compositionality at the syntax-semantics interface”. In: *J. Language Modelling* 6.1, pp. 41–86. DOI: 10.15398/jlm.v6i1.138. URL: https://doi.org/10.15398/jlm.v6i1.138.

Brants, Thorsten (2000). “Inter-annotator Agreement for a German Newspaper Corpus”. In: *Proceedings of the Second International Conference on Language*

Resources and Evaluation, LREC 2000, 31 May - June 2, 2000, Athens, Greece. URL: http://www.lrec-conf.org/proceedings/lrec2000/pdf/333.pdf.

Brochu, Eric, Vlad M. Cora, and Nando de Freitas (2010). "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning". In: *CoRR* abs/1012.2599. arXiv: 1012.2599. URL: http://arxiv.org/abs/1012.2599.

Burchardt, Aljoscha et al. (2006). "The SALSA Corpus: a German Corpus Resource for Lexical Semantics". In: *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06).* Genoa, Italy: European Language Resources Association (ELRA). URL: http://www.lrec-conf.org/proceedings/lrec2006/pdf/339_pdf.pdf.

Burghardt, Manuel, Julian Pörsch, Bianca Tirlea, and Christian Wolff (2014). "WebNLP - An Integrated Web-Interface for Python NLTK and Voyant". In: *Proceedings of the 12th Edition of the Konvens Conference, Hildesheim, Germany, October 8-10, 2014*, pp. 235–240.

"TEI P5: Guidelines for Electronic Text Encoding and Interchange" (2007). In: ed. by Lou Burnard and Syd Bauman. Text Encoding Initiative Consortium. Chap. A Gentle Introduction to XML. URL: http://www.tei-c.org/release/doc/tei-p5-doc/en/html/SG.html.

Carletta, Jean, Jonathan Kilgour, Tim O'Donnell, and Tim O'donnell (2003). "The NITE Object Model Library for Handling Structured Linguistic Annotation on Multimodal Data Sets". In: *In Proceedings of the EACL Workshop on Language Technology and the Semantic Web (3rd Workshop on NLP and XML, NLPXML-2003*, p. 2003.

Chaplot, Devendra Singh and Ruslan Salakhutdinov (2018). "Knowledge-based Word Sense Disambiguation using Topic Models". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 5062–5069. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17415.

Chifu, Adrian-Gabriel, Florentina Hristea, Josiane Mothe, and Marius Popescu (2015). "Word sense discrimination in information retrieval: A spectral clustering-based approach". In: *Inf. Process. Manage.* 51.2, pp. 16–31. DOI: 10.1016/j.ipm.2014.10.007. URL: https://doi.org/10.1016/j.ipm.2014.10.007.

Claesen, Marc and Bart De Moor (2015). "Hyperparameter Search in Machine Learning". In: *CoRR* abs/1502.02127. arXiv: 1502.02127. URL: http://arxiv.org/abs/1502.02127.

Conlon, Sumali J., Jason G. Hale, Susan Lukose, and Jody Strong (2008). "Information Extraction Agents for Service-Oriented Architecture Using Web Service Systems: A Framework". In: *JCIS* 48.3, pp. 74–83. URL: https://www.tandfonline.com/doi/abs/10.1080/08874417.2008.11646023.

Copeland, R. (2013). *MongoDB Applied Design Patterns: Practical Use Cases with the Leading NoSQL Database*. O'Reilly Media.

Crossley, Scott A. et al. (2015). "Language to Completion: Success in an Educational Data Mining Massive Open Online Class". In: *Proceedings of the 8th International Conference on Educational Data Mining, EDM 2015, Madrid, Spain, June 26-29, 2015*, pp. 388–391. URL: http://www.educationaldatamining.org/EDM2015/proceedings/short388-391.pdf.

Cunningham, Hamish et al. (2011). *Text Processing with GATE (Version 6)*. ISBN: 978-0956599315. URL: http://tinyurl.com/gatebook.

Dahl, Östen (2016). "Thoughts on language-specific and crosslinguistic entities". In: *Linguistic Typology* 20, pp. 427–437.

Dayley, Brad (2014). *NoSQL with MongoDB in 24 Hours, Sams Teach Yourself*. 1st. Sams publishing.

Deerwester, Scott C., Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman (1990). "Indexing by Latent Semantic Analysis". In: *JASIS* 41.6, pp. 391–407. DOI: 10.1002/(SICI)1097-4571(199009)41:6\<391::AID-ASI1\>3.0.CO;2-9. URL: https://doi.org/10.1002/(SICI)1097-4571(199009)41:6%5C%3C391::AID-ASI1%5C%3E3.0.CO;2-9.

Desel, Jörg and Wolfgang Reisig (1996). "Place/transition Petri nets". In: *Advanced Course on Petri Nets*. Springer, pp. 122–173.

Desjardins, Jeff (2019). *How much data is generated each day? | World Economic Forum*. https://www.weforum.org/agenda/2019/04/how-much-data-is-generated-each-day-cf4bddf29f/. (Accessed on 11/04/2019).

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805*.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. URL: `https://www.aclweb.org/anthology/N19-1423/`.

Dietterich, Thomas G. (2000). "Ensemble Methods in Machine Learning". In: *Multiple Classifier Systems, First International Workshop, MCS 2000, Cagliari, Italy, June 21-23, 2000, Proceedings*, pp. 1–15. DOI: `10.1007/3-540-45014-9\_1`. URL: `https://doi.org/10.1007/3-540-45014-9%5C_1`.

Duden, Konrad, Dieter Berger, and Werner Scholze (1980). *Duden*. Vol. 2. Bibliographisches Institut.

Eckart de Castilho, Richard and Iryna Gurevych (2014). "A broad-coverage collection of portable NLP components for building shareable analysis pipelines". In: *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT, OIAF4HLT@COLING 2014, Dublin, Ireland, August 23, 2014*, pp. 1–11. DOI: `10.3115/v1/W14-5201`. URL: `https://doi.org/10.3115/v1/W14-5201`.

Edmonds, Philip and Scott Cotton (2001). "SENSEVAL-2: Overview". In: *Proceedings of Second International Workshop on Evaluating Word Sense Disambiguation Systems, SENSEVAL@ACL 2001, Toulouse, France, July 5-6, 2001*, pp. 1–5. URL: `https://aclanthology.info/papers/S01-1001/s01-1001`.

Eger, Steffen and Alexander Mehler (2016). "On the Linearity of Semantic Change: Investigating Meaning Variation via Dynamic Graph Models". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. URL: `https://www.aclweb.org/anthology/P16-2009/`.

Eggert, Lisa, Maximilian Lippert, and Fabian Etling (2019). “Tagungsbericht Ännotationen in Edition und Forschung. Funktionsbestimmung, Differenzierung und Systematisierunğ”. In:

Elman, Jeffrey L. (1990). “Finding Structure in Time”. In: *Cognitive Science* 14.2, pp. 179–211. DOI: `10.1207/s15516709cog1402\_1`. URL: `https://doi.org/10.1207/s15516709cog1402%5C_1`.

Emmert-Streib, Frank, Matthias Dehmer, and Benjamin Haibe-Kains (2014). “Gene regulatory networks and their applications: understanding biological and medical problems in terms of networks”. In: *Frontiers in cell and developmental biology* 2, p. 38.

Erl, Thomas (2016). *SOA Principles of Service Design (Paperback).* 1st. Upper Saddle River, NJ, USA: Prentice Hall Press.

Fellbaum, C. and G. Miller (1998). “Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms”. In: *WordNet: An Electronic Lexical Database.* MITP, pp. 305–332. ISBN: 9780262272551. URL: `https://ieeexplore.ieee.org/document/6287673`.

Fellbaum, Christiane, Martha Palmer, Hoa Trang Dang, Lauren Delfs, and Steven A. Wolf (2001). “Manual and automatic semantic annotation with WordNet”. In:

Ferrucci, David A. and Adam Lally (2004). “UIMA: an architectural approach to unstructured information processing in the corporate research environment”. In: *Natural Language Engineering* 10.3-4, pp. 327–348. DOI: `10.1017/S1351324904003523`. URL: `https://doi.org/10.1017/S1351324904003523`.

Fette, Georg, Martin Toepfer, and Frank Puppe (2013). “Storing UIMA CASes in a relational database”. In: *Proceedings of the 3rd Workshop on Unstructured Information Management Architecture, Darmstadt, Germany, September 23, 2013*, pp. 10–13. URL: `http://ceur-ws.org/Vol-1038/paper%5C_1.pdf`.

Finkel, Jenny Rose, Shipra Dingare, et al. (2005). “Exploring the boundaries: gene and protein identification in biomedical text”. In: *BMC Bioinformatics* 6.S-1. DOI: `10.1186/1471-2105-6-S1-S5`. URL: `https://doi.org/10.1186/1471-2105-6-S1-S5`.

Finkel, Jenny Rose, Trond Grenager, and Christopher D. Manning (2005). “Incorporating Non-local Information into Information Extraction Systems by Gibbs

Sampling”. In: *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pp. 363–370. URL: `https://www.aclweb.org/anthology/P05-1045/`.

Ganesan, K., C. Zhai, and J. Han (2010). “Opinosis: A Graph-based Approach to Abstractive Summarization of Highly Redundant Opinions”. In: *Proc. of COLING ’10*, pp. 340–348.

Gardner, Matt et al. (2018). “AllenNLP: A Deep Semantic Natural Language Processing Platform”. In: *CoRR* abs/1803.07640. arXiv: `1803.07640`. URL: `http://arxiv.org/abs/1803.07640`.

Gawehn, Erik, Jan A. Hiss, and Gisbert Schneider (2016). “Deep Learning in Drug Discovery”. In: *Molecular Informatics* 35.1, pp. 3–14.

Geyer, Kelly, Kara Greenfield, Alyssa Mensch, and Olga Simek (2016). “Named Entity Recognition in 140 Characters or Less”. In: *Proceedings of the 6th Workshop on 'Making Sense of Microposts' co-located with the 25th International World Wide Web Conference (WWW 2016), Montréal, Canada, April 11, 2016.* Pp. 78–79. URL: `http://ceur-ws.org/Vol-1691/paper%5C_16.pdf`.

Giuglea, Ana-Maria and Alessandro Moschitti (2006). “Shallow Semantic Parsing Based on FrameNet, VerbNet and PropBank”. In: *ECAI 2006, 17th European Conference on Artificial Intelligence, August 29 - September 1, 2006, Riva del Garda, Italy, Including Prestigious Applications of Intelligent Systems (PAIS 2006), Proceedings*, pp. 563–567.

Gleim, Rüdiger, Alexander Mehler, and Alexandra Ernst (2012). “SOA implementation of the eHumanities Desktop”. In: *Proceedings of the Workshop on Service-oriented Architectures (SOAs) for the Humanities: Solutions and Impacts, Digital Humanities 2012, Hamburg, Germany.*

Golub, David and Xiaodong He (2016). “Character-Level Question Answering with Attention”. In: *CoRR* abs/1604.00727. arXiv: `1604.00727`. URL: `http://arxiv.org/abs/1604.00727`.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey E. Hinton (2013). “Speech Recognition with Deep Recurrent Neural Networks”. In: *CoRR* abs/1303.5778. arXiv: `1303.5778`. URL: `http://arxiv.org/abs/1303.5778`.

Gritzmann, Peter (2007). “On the Mathematics of Semantic Spaces”. In: *Aspects of Automatic Text Analysis*, pp. 95–115. DOI: 10.1007/978-3-540-37522-7_5. URL: https://doi.org/10.1007/978-3-540-37522-7%5C_5.

Grose, T., G. Doney, and S. Brodsky (2002). *Mastering XMI: Java Programming with XMI, XML and UML*. Vol. 21. John Wiley & Sons.

Hahn, Udo et al. (2008). “An overview of JCoRe, the JULIE lab UIMA component repository”. In: *Proc. of the LREC*. Vol. 8, pp. 1–7.

Hammerton, James (2003). “Named Entity Recognition with Long Short-Term Memory”. In: *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pp. 172–175. URL: https://www.aclweb.org/anthology/W03-0426/.

Hamp, Birgit and Helmut Feldweg (1997). “GermaNet - a Lexical-Semantic Net for German”. In: *In Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pp. 9–15.

Hand, David J (2006). “Data Mining”. In: *Encyclopedia of Environmetrics* 2.

Hapner, Mark, Rich Burridge, Rahul Sharma, Joseph Fialli, and Kate Stout (2002). “Java message service”. In: *Sun Microsystems Inc., Santa Clara, CA* 9.

Harris, Zellig S (1954). “Distributional structure”. In: *Word* 10.2-3, pp. 146–162.

Hemati, Wahed and Alexander Mehler (2019a). “CRFVoter: gene and protein related object recognition using a conglomerate of CRF-based tools”. In: *J. Cheminformatics* 11.1, 21:1–21:11. DOI: 10.1186/s13321-019-0343-x. URL: https://doi.org/10.1186/s13321-019-0343-x.

– (2019b). “LSTMVoter: chemical named entity recognition using a conglomerate of sequence labeling tools”. In: *J. Cheminformatics* 11.1, 3:1–3:7. DOI: 10.1186/s13321-018-0327-2. URL: https://doi.org/10.1186/s13321-018-0327-2.

Hemati, Wahed, Alexander Mehler, and Tolga Uslu (2017). “CRFVoter: Chemical Entity Mention, Gene and Protein Related Object recognition using a conglomerate of CRF based tools”. In: *BioCreative V.5. Proceedings*.

Hemati, Wahed, Alexander Mehler, Tolga Uslu, and Giuseppe Abrami (2019). “Der TextImager als Front- und Backend für das verteilte NLP von Big Digital

Humanities Data". In: *Proceedings of the 6th Digital Humanities Conference in the German-speaking Countries, DHd 2019*. DHd 2019. Frankfurt, Germany.

Hemati, Wahed, Alexander Mehler, Tolga Uslu, Daniel Baumartz, and Giuseppe Abrami (2018). "Evaluating and Integrating Databases in the Area of NLP". In: *International Quantitative Linguistics Conference (QUALICO 2018)*. Wroclaw, Poland.

Hemati, Wahed, Tolga Uslu, and Alexander Mehler (2016). "TextImager: a Distributed UIMA-based System for NLP". In: *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference System Demonstrations, December 11-16, 2016, Osaka, Japan*, pp. 59–63. URL: `https://www.aclweb.org/anthology/C16-2013/`.

– (2017). "TextImager as an interface to BeCalm". In: *BioCreative V.5. Proceedings.*

Henrich, Verena (2015). "Word Sense Disambiguation with GermaNet". en. PhD thesis. DOI: `10.15496/publikation-4706`. URL: `https://publikationen.uni-tuebingen.de/xmlui/handle/10900/63284`.

Henrich, Verena and Erhard W. Hinrichs (2013). "Extending the TüBa-D/Z Treebank with GermaNet Sense Annotation". In: *Language Processing and Knowledge in the Web - 25th International Conference, GSCL 2013, Darmstadt, Germany, September 25-27, 2013. Proceedings*, pp. 89–96. DOI: `10.1007/978-3-642-40722-2\_9`. URL: `https://doi.org/10.1007/978-3-642-40722-2%5C_9`.

Henrich, Verena, Erhard W. Hinrichs, and Tatiana Vodolazova (2011). "Aligning GermaNet Senses with Wiktionary Sense Definitions". In: *Human Language Technology Challenges for Computer Science and Linguistics - 5th Language and Technology Conference, LTC 2011, Poznań, Poland, November 25-27, 2011, Revised Selected Papers*, pp. 329–342. DOI: `10.1007/978-3-319-08958-4\_27`. URL: `https://doi.org/10.1007/978-3-319-08958-4%5C_27`.

– (2012). "WebCAGe - A Web-Harvested Corpus Annotated with GermaNet Senses". In: *EACL 2012, 13th Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, April 23-27, 2012*, pp. 387–396. URL: `http://aclweb.org/anthology/E/E12/E12-1039.pdf`.

Hewitt, Eben (2011). *Cassandra - The Definitive Guide: Distributed Data at Web Scale*. Springer. ISBN: 978-1-449-39041-9. URL: `http://www.oreilly.de/catalog/9781449390419/index.html`.

Hinrichs, Erhard W., Marie Hinrichs, and Thomas Zastrow (2010). "WebLicht: Web-Based LRT Services for German". In: *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, System Demonstrations*, pp. 25–29. URL: `https://www.aclweb.org/anthology/P10-4005/`.

Hinrichs, Marie, Thomas Zastrow, and Erhard W. Hinrichs (2010). "WebLicht: Web-based LRT Services in a Distributed eScience Infrastructure". In: *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta.* URL: `http://www.lrec-conf.org/proceedings/lrec2010/summaries/270.html`.

Hirschman, Lynette, Alexander S. Yeh, Christian Blaschke, and Alfonso Valencia (2005). "Overview of BioCreAtIvE: critical assessment of information extraction for biology". In: *BMC Bioinformatics* 6.S-1. DOI: `10.1186/1471-2105-6-S1-S1`. URL: `https://doi.org/10.1186/1471-2105-6-S1-S1`.

Hochreiter, Sepp (1998). "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.2, pp. 107–116. DOI: `10.1142/S0218488598000094`. URL: `https://doi.org/10.1142/S0218488598000094`.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780. DOI: `10.1162/neco.1997.9.8.1735`. URL: `https://doi.org/10.1162/neco.1997.9.8.1735`.

Hofmann, Markus and Ralf Klinkenberg (2013). *RapidMiner: Data mining use cases and business analytics applications*. CRC Press.

Honnibal, Matthew and Ines Montani (2017). "spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing". In: *To appear* 7.

Hung, Jason C., Ching-Sheng Wang, Che-Yu Yang, Mao-Shuen Chiu, and George Yee (2005). "Applying Word Sense Disambiguation to Question Answering System for e-Learning". In: *19th International Conference on Advanced Information Networking and Applications (AINA 2005), 28-30 March 2005, Taipei,* *Taiwan*, pp. 157–162. DOI: 10.1109/AINA.2005.121. URL: https://doi.org/10.1109/AINA.2005.121.

Hunziker, Alex, Hasanagha Mammadov, Wahed Hemati, and Alexander Mehler (2019). "Corpus2Wiki: A MediaWiki-based Tool for Automatically Generating Wikiditions in Digital Humanities". In: *INFORMATIK 2019: 50 Jahre Gesellschaft für Informatik–Informatik für Gesellschaft (Workshop-Beiträge).* Gesellschaft für Informatik eV.

Hutter, Frank, Holger H. Hoos, and Kevin Leyton-Brown (2011). "Sequential Model-Based Optimization for General Algorithm Configuration". In: *Learning and Intelligent Optimization - 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers*, pp. 507–523. DOI: 10.1007/978-3-642-25566-3_40. URL: https://doi.org/10.1007/978-3-642-25566-3%5C_40.

– (2014). "An Efficient Approach for Assessing Hyperparameter Importance". In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 754–762. URL: http://proceedings.mlr.press/v32/hutter14.html.

Jänicke, Stefan, Greta Franzini, Muhammad Faisal Cheema, and Gerik Scheuermann (2015). "On Close and Distant Reading in Digital Humanities: A Survey and Future Challenges". In: *Eurographics Conference on Visualization, EuroVis 2015 - State of the Art Reports, STARs, Cagliari, Italy, May 25-29, 2015*, pp. 83–103. DOI: 10.2312/eurovisstar.20151113. URL: https://doi.org/10.2312/eurovisstar.20151113.

Jiang, Jay J. and David W. Conrath (1997). "Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy". In: *Proceedings of the 10th Research on Computational Linguistics International Conference, ROCLING 1997, Taipei, Taiwan, August 1997*, pp. 19–33. URL: https://aclanthology.info/papers/O97-1002/o97-1002.

Kett, Attila, Giuseppe Abrami, Alexander Mehler, and Christian Spiekermann (2018). "resources2city Explorer: A System for Generating Interactive Walkable Virtual Cities out of File Systems". In: *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings, UIST 2018,* *Berlin, Germany, October 14-17, 2018*, pp. 123–125. DOI: 10.1145/3266037.3266122. URL: https://doi.org/10.1145/3266037.3266122.

Khabsa, Madian and C. Lee Giles (2015). "Chemical entity extraction using CRF and an ensemble of extractors". In: *J. Cheminformatics* 7.S-1, S12. DOI: 10.1186/1758-2946-7-S1-S12. URL: https://doi.org/10.1186/1758-2946-7-S1-S12.

Khalili, Ali, Sören Auer, and Axel-Cyrille Ngonga Ngomo (2014). "conTEXT - Lightweight Text Analytics Using Linked Data". In: *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*, pp. 628–643. DOI: 10.1007/978-3-319-07443-6_42. URL: https://doi.org/10.1007/978-3-319-07443-6%5C_42.

Kilgarriff, Adam (1997). ""I Don't Believe in Word Senses"". In: *Computers and the Humanities* 31.2, pp. 91–113. DOI: 10.1023/A:1000583911091. URL: https://doi.org/10.1023/A:1000583911091.

– (1998). "Gold standard datasets for evaluating word sense disambiguation programs". In: *Computer Speech & Language* 12.4, pp. 453–472. DOI: 10.1006/csla.1998.0108. URL: https://doi.org/10.1006/csla.1998.0108.

– (1999). "95% Replicability for Manual Word Sense Tagging". In: *EACL 1999, 9th Conference of the European Chapter of the Association for Computational Linguistics, June 8-12, 1999, University of Bergen, Bergen, Norway*, pp. 277–278. URL: http://aclweb.org/anthology/E/E99/E99-1046.pdf.

Kim, Yoon, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov (2014). "Temporal Analysis of Language through Neural Language Models". In: *CoRR* abs/1405.3515. arXiv: 1405.3515. URL: http://arxiv.org/abs/1405.3515.

Komninos, Alexandros and Suresh Manandhar (2016). "Dependency Based Embeddings for Sentence Classification Tasks". In: *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pp. 1490–1500. URL: https://www.aclweb.org/anthology/N16-1175/.

Köper, Maximilian, Sabine Schulte im Walde, Max Kisselew, and Sebastian Padó (2016). "Improving Zero-Shot-Learning for German Particle Verbs by using Training-Space Restrictions and Local Scaling". In: *Proceedings of the Fifth*

*Joint Conference on Lexical and Computational Semantics, *SEM@ACL 2016, Berlin, Germany, 11-12 August 2016.* URL: https://www.aclweb.org/anthology/S16-2010/.

Krallinger, Martin, Florian Leitner, et al. (2015). "CHEMDNER: The drugs and chemical names extraction challenge". In: *J. Cheminformatics* 7.S-1, S1. DOI: 10.1186/1758-2946-7-S1-S1. URL: https://doi.org/10.1186/1758-2946-7-S1-S1.

Krallinger, Martin, Martin Pérez-Pérez, et al. (2017). "The BioCreative V.5 evaluation workshop: tasks, organization, sessions and topics". In: *Proceedings of the BioCreative V.5 Challenge Evaluation Workshop*, pp. 8–10.

Krallinger, Martin, Obdulia Rabal, et al. (2015). "Overview of the CHEMDNER patents task". In: *Proceedings of the 5th BioCreative Challenge Evaluation Workshop.*

Krallinger, Martin, Miguel Vazquez, et al. (2011). "The Protein-Protein Interaction tasks of BioCreative III: classification/ranking of articles and linking bio-ontology concepts to full text". In: *BMC Bioinformatics* 12.S-8, S3. DOI: 10.1186/1471-2105-12-S8-S3. URL: https://doi.org/10.1186/1471-2105-12-S8-S3.

Krippendorff, Klaus (2018). *Content analysis: An introduction to its methodology.* Sage publications.

Kuczera, Andreas (2016). "Digital Editions beyond XML - Graph-based Digital Editions". In: *Proceedings of the 3rd HistoInformatics Workshop on Computational History (HistoInformatics 2016) co-located with Digital Humanities 2016 conference (DH 2016), Krakow, Poland, July 11, 2016.* Pp. 37–46. URL: http://ceur-ws.org/Vol-1632/paper%5C_5.pdf.

Kudo, Taku (2005). "CRF++: Yet another CRF toolkit". In: *Software available at https://taku910.github.io/crfpp/.*

Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira (2001). "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pp. 282–289.

Lample, Guillaume, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer (2016). “Neural Architectures for Named Entity Recognition”. In: *CoRR* abs/1603.01360. arXiv: 1603.01360. URL: http://arxiv.org/abs/1603.01360.

Lavrentiev, Alexei, Dominique Stutzmann, and Yann Leydier (2015). “Specifying a TEI-XML Based Format for Aligning Text to Image at Character Level”. In: *Symposium on Cultural Heritage Markup*. Vol. 16, BalisageVol16–Lavrentiev01.

Leacock, Claudia and Martin Chodorow (1998). “Combining local context and WordNet similarity for word sense identification”. In: *WordNet: An electronic lexical database* 49.2, pp. 265–283.

Lesk, Michael (1986). “Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone”. In: *Proceedings of the 5th Annual International Conference on Systems Documentation, SIGDOC 1986, Toronto, Ontario, Canada, 1986*, pp. 24–26. DOI: 10.1145/318723.318728. URL: https://doi.org/10.1145/318723.318728.

Levin, Beth (1993). *English verb classes and alternations:a preliminary investigation*. University of Chicago Press.

Levy, Omer and Yoav Goldberg (2014). “Dependency-Based Word Embeddings”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pp. 302–308. URL: https://www.aclweb.org/anthology/P14-2050/.

Leydier, Yann, Véronique Eglin, Stéphane Bres, and Dominique Stutzmann (2014). “Learning-Free Text-Image Alignment for Medieval Manuscripts”. In: *14th International Conference on Frontiers in Handwriting Recognition, ICFHR 2014, Crete, Greece, September 1-4, 2014*, pp. 363–368. DOI: 10.1109/ICFHR.2014.67. URL: https://doi.org/10.1109/ICFHR.2014.67.

Li, Jiao et al. (2016). “BioCreative V CDR task corpus: a resource for chemical disease relation extraction”. In: *Database* 2016. DOI: 10.1093/database/baw068. URL: https://doi.org/10.1093/database/baw068.

Ling, Wang, Chris Dyer, Alan W. Black, and Isabel Trancoso (2015). “Two/Too Simple Adaptations of Word2Vec for Syntax Problems”. In: *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015, pp. 1299–1304. URL: https://www.aclweb.org/anthology/N15-1142/.

Lu, Yanan, Donghong Ji, Xiaoyuan Yao, Xiaomei Wei, and Xiaohui Liang (2015). "CHEMDNER system with mixed conditional random fields and multi-scale word clustering". In: *J. Cheminformatics* 7.S-1, S4. DOI: 10.1186/1758-2946-7-S1-S4. URL: https://doi.org/10.1186/1758-2946-7-S1-S4.

Lu, Zhiyong et al. (2011). "The gene normalization task in BioCreative III". In: *BMC Bioinformatics* 12.S-8, S2. DOI: 10.1186/1471-2105-12-S8-S2. URL: https://doi.org/10.1186/1471-2105-12-S8-S2.

Luo, Fuli, Tianyu Liu, Qiaolin Xia, Baobao Chang, and Zhifang Sui (2018). "Incorporating Glosses into Neural Word Sense Disambiguation". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pp. 2473–2482. URL: https://aclanthology.info/papers/P18-1230/p18-1230.

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning (2015). "Effective Approaches to Attention-based Neural Machine Translation". In: *CoRR* abs/1508.04025. arXiv: 1508.04025. URL: http://arxiv.org/abs/1508.04025.

Lyon, W. (2016). *Natural Language Processing with Graph Databases and Neo4j.*

Manning, Christopher D. et al. (2014). "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*, pp. 55–60. URL: https://www.aclweb.org/anthology/P14-5010/.

Maurizio, Amelia A., James Sager, Peter Jones, Gail Corbitt, and Louis Girolami (2008). "Service Oriented Architecture: Challenges for Business and Academia". In: *41st Hawaii International International Conference on Systems Science (HICSS-41 2008), Proceedings, 7-10 January 2008, Waikoloa, Big Island, HI, USA*, p. 315. DOI: 10.1109/HICSS.2008.387. URL: https://doi.org/10.1109/HICSS.2008.387.

McCann, Bryan, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher (2018). “The Natural Language Decathlon: Multitask Learning as Question Answering”. In: *CoRR* abs/1806.08730. arXiv: 1806.08730. URL: http://arxiv.org/abs/1806.08730.

McNemar, Quinn (1947). “Note on the sampling error of the difference between correlated proportions or percentages”. In: *Psychometrika* 12.2, pp. 153–157.

Mehler, Alexander (2005). “Eigenschaften der textuellen Einheiten und Systeme (Properties of textual units and systems)”. In: *Quantitative Linguistik / Quantitative Linguistics - Ein internationales Handbuch / An International Handbook*, pp. 325–347.

Mehler, Alexander, Giuseppe Abrami, Christian Spiekermann, and Matthias Jostock (2018). “VAnnotatoR: A Framework for Generating Multimodal Hypertexts”. In: *Proceedings of the 29th on Hypertext and Social Media, HT 2018, Baltimore, MD, USA, July 09-12, 2018*, pp. 150–154. DOI: 10.1145/3209542.3209572. URL: https://doi.org/10.1145/3209542.3209572.

Mehler, Alexander, Rüdiger Gleim, Tim vor der Brück, et al. (2016). “Wikidition: Automatic lexiconization and linkification of text corpora”. In: *it - Information Technology* 58.2, pp. 70–79. URL: http://www.degruyter.com/view/j/itit.2016.58.issue-2/itit-2015-0035/itit-2015-0035.xml.

Mehler, Alexander, Rüdiger Gleim, Ulli Waltinger, et al. (2009). “eHumanities Desktop – eine webbasierte Arbeitsumgebung für die geisteswissenschaftliche Fachinformatik”. In: *Proceedings of the Symposium ”Sprachtechnologie und eHumanities”, 26.–27. Februar, Duisburg-Essen University.*

Mehler, Alexander, Wahed Hemati, Rüdiger Gleim, and Daniel Baumartz (2018). “VienNA: Auf dem Weg zu einer Infrastruktur für die verteilte interaktive evolutionäre Verarbeitung natürlicher Sprache”. In: *Forschungsinfrastrukturen und digitale Informationssysteme in der germanistischen Sprachwissenschaft.* Ed. by Henning Lobin, Roman Schneider, and Andreas Witt. Vol. 6. Berlin: De Gruyter.

Mehler, Alexander, Wahed Hemati, Tolga Uslu, and Andy Lücking (2018). “A Multidimensional Model of Syntactic Dependency Trees for Authorship Attribution”. In: *Quantitative analysis of dependency structures.* Ed. by Jingyang Jiang and Haitao Liu. Berlin/New York: De Gruyter.

Mehler, Alexander and Reinhard Köhler, eds. (2007). *Aspects of Automatic Text Analysis*. Vol. 209. Studies in Fuzziness and Soft Computing. ISBN: 978-3-540-37520-3. DOI: 10.1007/978-3-540-37522-7. URL: https://doi.org/10.1007/978-3-540-37522-7.

Mehler, Alexander, Silke Schwandt, Rüdiger Gleim, and Bernhard Jussen (2011). "Der eHumanities Desktop als Werkzeug in der historischen Semantik: Funktionsspektrum und Einsatzszenarien". In: *JLCL* 26.1, pp. 97–117. URL: http://media.dwds.de/jlcl/2011%5C_Heft1/8.pdf.

Mehler, Alexander, Tolga Uslu, and Wahed Hemati (2016). "Text2voronoi: An Image-driven Approach to Differential Diagnosis". In: *Proceedings of the 5th Workshop on Vision and Language, hosted by the 54th Annual Meeting of the Association for Computational Linguistics, VL@ACL 2016, August 12, Berlin, Germany*. URL: https://www.aclweb.org/anthology/W16-3212/.

Mehler, Alexander, Benno Wagner, and Rüdiger Gleim (2016). "Wikidition: Towards A Multi-layer Network Model of Intertextuality". In: *Digital Humanities 2016, DH 2016, Conference Abstracts, Jagiellonian University & Pedagogical University, Krakow, Poland, July 11-16, 2016*, pp. 276–279. URL: http://dh2016.adho.org/abstracts/250.

Melamud, Oren, Jacob Goldberger, and Ido Dagan (2016). "context2vec: Learning Generic Context Embedding with Bidirectional LSTM". In: *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pp. 51–61. URL: http://aclweb.org/anthology/K/K16/K16-1006.pdf.

Michel, Jean-Baptiste et al. (2010). "Quantitative Analysis of Culture Using Millions of Digitized Books". In: *Science*. URL: http://www.sciencemag.org/content/331/6014/176.full.

Mihalcea, R. and P. Tarau (2004). "TextRank: Bringing Order into Texts". In: *Proc. of EMNLP-04*.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Efficient Estimation of Word Representations in Vector Space". In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. URL: http://arxiv.org/abs/1301.3781.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *CoRR* abs/1310.4546. arXiv: 1310.4546. URL: http://arxiv.org/abs/1310.4546.

Miller, George A. (1995). "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11, pp. 39–41. DOI: 10.1145/219717.219748. URL: http://doi.acm.org/10.1145/219717.219748.

Miller, J. J. (2013). "Graph database applications and concepts with Neo4j". In: *Proc. SAIS 2013*, p. 36.

Morgan, Alexander A. et al. (2008). "Overview of BioCreative II gene normalization". In: *Genome Biology* 9.2, S3. ISSN: 1474-760X. DOI: 10.1186/gb-2008-9-s2-s3.

Moro, Andrea, Alessandro Raganato, and Roberto Navigli (2014). "Entity Linking meets Word Sense Disambiguation: a Unified Approach". In: *TACL* 2, pp. 231–244. URL: https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/291.

Mu, Jesse, Joshua K. Hartshorne, and Timothy O'Donnell (2017). "Evaluating Hierarchies of Verb Argument Structure with Hierarchical Clustering". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pp. 986–991. URL: https://www.aclweb.org/anthology/D17-1104/.

Müller, Bernd and Alexandra Hagelstein (2016). "Beyond Metadata: Enriching life science publications in Livivo with semantic entities from the linked data cloud". In: *Joint Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems - SEMANTiCS2016 and the 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS'16) co-located with the 12th International Conference on Semantic Systems (SEMANTiCS 2016), Leipzig, Germany, September 12-15, 2016.* URL: http://ceur-ws.org/Vol-1695/paper1.pdf.

Müller, Thomas, Helmut Schmid, and Hinrich Schütze (2013). "Efficient Higher-Order CRFs for Morphological Tagging". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of*

SIGDAT, a Special Interest Group of the ACL, pp. 322–332. URL: https://www.aclweb.org/anthology/D13-1032/.

Navigli, Roberto (2009). "Word sense disambiguation: A survey". In: *ACM Comput. Surv.* 41.2, 10:1–10:69. DOI: 10.1145/1459352.1459355. URL: https://doi.org/10.1145/1459352.1459355.

Navigli, Roberto, David Jurgens, and Daniele Vannella (2013). "SemEval-2013 Task 12: Multilingual Word Sense Disambiguation". In: *Proceedings of the 7th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2013, Atlanta, Georgia, USA, June 14-15, 2013*, pp. 222–231. URL: http://aclweb.org/anthology/S/S13/S13-2040.pdf.

Neale, Steven, Luís Gomes, Eneko Agirre, Oier Lopez de Lacalle, and António Branco (2016). "Word Sense-Aware Machine Translation: Including Senses as Contextual Features for Improved Translation Models". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.* URL: http://www.lrec-conf.org/proceedings/lrec2016/summaries/1078.html.

Newman, Mark EJ (2005). "Power laws, Pareto distributions and Zipf's law". In: *Contemporary physics* 46.5, pp. 323–351.

Ng, Hwee Tou (2011). "Does word sense disambiguation improve information retrieval?" In: *Proceedings of the fourth workshop on Exploiting Semantic Annotations in Information Retrieval, ESAIR 2011, Glasgow, United Kingdom, October 28, 2011*, pp. 17–18. DOI: 10.1145/2064713.2064724. URL: https://doi.org/10.1145/2064713.2064724.

OpenNLP (2010). *Apache OpenNLP, http://opennlp.apache.org*. URL: http://opennlp.apache.org.

Ott, Myle et al. (2019). "fairseq: A Fast, Extensible Toolkit for Sequence Modeling". In: *Proceedings of NAACL-HLT 2019: Demonstrations.*

Pal, Alok Ranjan and Diganta Saha (2015). "Word sense disambiguation: a survey". In: *CoRR* abs/1508.01346. arXiv: 1508.01346. URL: http://arxiv.org/abs/1508.01346.

Palmer, Martha, Hoa Trang Dang, and Christiane Fellbaum (2007). "Making fine-grained and coarse-grained sense distinctions, both manually and automatically". In: *Natural Language Engineering* 13.2, pp. 137–163. DOI: 10.1017/S135132490500402X. URL: https://doi.org/10.1017/S135132490500402X.

Palmer, Martha, Christiane Fellbaum, Scott Cotton, Lauren Delfs, and Hoa Trang Dang (2001). "English Tasks: All-Words and Verb Lexical Sample". In: *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*. Toulouse, France: Association for Computational Linguistics, pp. 21–24. URL: https://www.aclweb.org/anthology/S01-1005.

Papandrea, Simone, Alessandro Raganato, and Claudio Delli Bovi (2017). "SupWSD: A Flexible Toolkit for Supervised Word Sense Disambiguation". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017 - System Demonstrations*, pp. 103–108. URL: https://aclanthology.info/papers/D17-2018/d17-2018.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2012). "Understanding the exploding gradient problem". In: *CoRR* abs/1211.5063. arXiv: 1211.5063. URL: http://arxiv.org/abs/1211.5063.

Passonneau, Rebecca J., Collin F. Baker, Christiane Fellbaum, and Nancy Ide (2012). "The MASC Word Sense Corpus". In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pp. 3025–3030. URL: http://www.lrec-conf.org/proceedings/lrec2012/summaries/589.html.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1532–1543. URL: https://www.aclweb.org/anthology/D14-1162/.

Pérez-Pérez, M et al. (2017). "Evaluation of chemical and gene/protein entity recognition systems at BioCreative V.5: the CEMP and GPRO patents tracks". In: *Proceedings of the BioCreative V.5 Challenge Evaluation Workshop*, pp. 11–18.

Perry, Steven (2017). *What is big data? More than volume, velocity and variety... - The developerWorks Blog*. https://developer.ibm.com/dwblog/2017/what-is-big-data-insight/. (Accessed on 11/04/2019).

Peters, Matthew E. et al. (2018a). "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pp. 2227–2237. URL: https://aclanthology.info/papers/N18-1202/n18-1202.

– (2018b). "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pp. 2227–2237. URL: https://www.aclweb.org/anthology/N18-1202/.

Plis, Sergey M., R. Devon Hjelm, Ruslan Salakhutdinov, and Vince D. Calhoun (2014). "Deep learning for neuroimaging: a validation study". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*. URL: http://arxiv.org/abs/1312.5847.

Popel, Martin and Zdenek Zabokrtský (2010). "TectoMT: Modular NLP Framework". In: *Advances in Natural Language Processing, 7th International Conference on NLP, IceTAL 2010, Reykjavik, Iceland, August 16-18, 2010*, pp. 293–304. DOI: 10.1007/978-3-642-14770-8_33. URL: https://doi.org/10.1007/978-3-642-14770-8%5C_33.

Pradhan, Sameer, Edward Loper, Dmitriy Dligach, and Martha Palmer (2007). "SemEval-2007 Task-17: English Lexical Sample, SRL and All Words". In: *Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval@ACL 2007, Prague, Czech Republic, June 23-24, 2007*, pp. 87–92. URL: http://aclweb.org/anthology/S/S07/S07-1016.pdf.

PubMed - NCBI. https://www.ncbi.nlm.nih.gov/pubmed/. Accessed 19 July 2018.

Qayyum, Adnan, Syed Muhammad Anwar, Muhammad Majid, Muhammad Awais, and Majdi R. Alnowami (2017). "Medical Image Analysis using Convolutional

Neural Networks: A Review". In: *CoRR* abs/1709.02250. arXiv: 1709.02250. URL: http://arxiv.org/abs/1709.02250.

Raileanu, Diana, Paul Buitelaar, Spela Vintar, and Jörg Bay (2002). "Evaluation Corpora for Sense Disambiguation in the Medical Domain". In: *Proceedings of the Third International Conference on Language Resources and Evaluation, LREC 2002, May 29-31, 2002, Las Palmas, Canary Islands, Spain.* URL: http://www.lrec-conf.org/proceedings/lrec2002/sumarios/166.htm.

Ramshaw, Lance A. and Mitchell P. Marcus (1995a). "Text Chunking using Transformation-Based Learning". In: *CoRR* cmp-lg/9505040. URL: http://arxiv.org/abs/cmp-lg/9505040.

– (1995b). "Text Chunking using Transformation-Based Learning". In: *CoRR* cmp-lg/9505040. URL: http://arxiv.org/abs/cmp-lg/9505040.

Randolph, Justus J (2005). "Free-Marginal Multirater Kappa (multirater K [free]): An Alternative to Fleiss' Fixed-Marginal Multirater Kappa." In: *Online submission.*

Rasmussen, Carl Edward (2003). "Gaussian Processes in Machine Learning". In: *Advanced Lectures on Machine Learning, ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tübingen, Germany, August 4-16, 2003, Revised Lectures*, pp. 63–71. DOI: 10.1007/978-3-540-28650-9_4. URL: https://doi.org/10.1007/978-3-540-28650-9%5C_4.

Rei, Marek, Gamal K. O. Crichton, and Sampo Pyysalo (2016). "Attending to Characters in Neural Sequence Labeling Models". In: *CoRR* abs/1611.04361. arXiv: 1611.04361. URL: http://arxiv.org/abs/1611.04361.

Reisig, Wolfgang (1985). *Petri Nets: An Introduction.* Vol. 4. EATCS Monographs on Theoretical Computer Science. Springer. ISBN: 3-540-13723-8. DOI: 10.1007/978-3-642-69968-9. URL: https://doi.org/10.1007/978-3-642-69968-9.

Resnik, Philip (1995). "Using Information Content to Evaluate Semantic Similarity in a Taxonomy". In: *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, pp. 448–453. URL: http://ijcai.org/Proceedings/95-1/Papers/059.pdf.

Reuber, Markus, Daniel J Blackburn, et al. (2018). “An interactional profile to assist the differential diagnosis of neurodegenerative and functional memory disorders”. In: *Alzheimer Disease & Associated Disorders* 32.3, pp. 197–206.

Reuber, Markus, Chiara Monzoni, Basil Sharrack, and Leendert Plug (2009). “Using interactional and linguistic analysis to distinguish between epileptic and psychogenic nonepileptic seizures: a prospective, blinded multirater study”. In: *Epilepsy & Behavior* 16.1, pp. 139–144.

Richardet, Renaud, Jean-Cédric Chappelier, and Martin Telefont (2013). “Bluima: a UIMA-based NLP Toolkit for Neuroscience”. In: *Proceedings of the 3rd Workshop on Unstructured Information Management Architecture, Darmstadt, Germany, September 23, 2013*, pp. 34–41. URL: http://ceur-ws.org/Vol-1038/paper%5C_7.pdf.

Richardson, Leonard and Sam Ruby (2007). *RESTful web services - web services for the real world.* O’Reilly. ISBN: 978-0-596-52926-0. URL: http://www.oreilly.com/catalog/9780596529260/index.html.

Rieger, Burghard (1995). “Situation Semantics and Computational Linguistics: towards Informational Ecology”. In: *Information. New Questions to a Multidisciplinary Concept.* Ed. by K. Kornwachs and K. Jacoby. Berlin: Akademie-Verlag, pp. 285–315.

Robinson, Carly, Michael Yeomans, Justin Reich, Chris Hulleman, and Hunter Gehlbach (2016). “Forecasting student achievement in MOOCs with natural language processing”. In: *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge, LAK 2016, Edinburgh, United Kingdom, April 25-29, 2016*, pp. 383–387. DOI: 10.1145/2883851.2883932. URL: https://doi.org/10.1145/2883851.2883932.

Rousseau, François, Emmanouil Kiagias, and Michalis Vazirgiannis (2015). “Text Categorization as a Graph Classification Problem”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pp. 1702–1712. URL: https://www.aclweb.org/anthology/P15-1164/.

Ruecker, Stan, Milena Radzikowska, and Stéfan Sinclair (2011). *Visual interface design for digital cultural heritage: A guide to rich-prospect browsing.* Ashgate Publishing, Ltd.

Rutherford, Eleanor, Wahed Hemati, and Alexander Mehler (2018). "Corpus2Wiki: A MediaWiki based Annotation & Visualisation Tool for the Digital Humanities". In: *GI-Workshop: Im Spannungsfeld zwischen Tool-Building und Forschung auf Augenhöhe - Informatik und die Digital Humanities, INF-DH 2018, 25.9.2018, Berlin, Germany.* DOI: `10.18420/infdh2018-08`. URL: `https://doi.org/10.18420/infdh2018-08`.

Sahami, Soheila, Thomas Eckart, and Gerhard Heyer (2019). "Using Apache Spark on Hadoop Clusters as Backend for WebLicht Processing Pipelines". In: *Selected papers from the CLARIN Annual Conference 2018, Pisa, 8-10 October 2018.* 159. Linköping University Electronic Press, pp. 183–190.

Saito, Jahn-Takeshi et al. (2003). "Evaluation of GermanNet : Problems Using GermaNet for Automatic Word Sense Disambiguation". In:

Sanders, Derek T., John A. Hamilton Jr., and Richard A. MacDonald (2008). "Supporting a service-oriented architecture". In: *Proceedings of the 2008 Spring Simulation Multiconference, SpringSim 2008, Ottawa, Canada, April 14-17, 2008*, pp. 325–334. URL: `http://dl.acm.org/citation.cfm?id=1400549.1400595`.

Sanh, Victor, Thomas Wolf, and Sebastian Ruder (2019). "A Hierarchical Multi-Task Approach for Learning Embeddings from Semantic Tasks". In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 6949–6956. DOI: `10.1609/aaai.v33i01.33016949`. URL: `https://doi.org/10.1609/aaai.v33i01.33016949`.

Schäfer, Roland (2015). "Processing and querying large web corpora with the COW14 architecture". In: *Proceedings of Challenges in the Management of Large Corpora 3 (CMLC-3).* UCREL. Lancaster: IDS. URL: `http://rolandschaefer.net/?p=749`.

Schäfer, Roland and Felix Bildhauer (2012). “Building Large Corpora from the Web Using a New Efficient Tool Chain”. In: *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12).* Istanbul, Turkey: European Language Resources Association (ELRA), pp. 486–493. ISBN: 978-2-9517408-7-7. URL: `http://rolandschaefer.net/?p=70`.

Scheible, Silke, Sabine Schulte im Walde, Marion Weller, and Max Kisselew (2013). “A compact but linguistically detailed database for German verb subcategorisation relying on dependency parses from Web corpora: Tool, guidelines and resource”. In: *Web as Corpus Workshop.*

Schuler, Karin Kipper (2006). “VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon”. PhD thesis. University of Pennsylvania. URL: `http://verbs.colorado.edu/~kipper/Papers/dissertation.pdf`.

Schulte im Walde, Sabine (2006). “Experiments on the Automatic Induction of German Semantic Verb Classes”. In: *Computational Linguistics* 32.2, pp. 159–194. DOI: `10.1162/coli.2006.32.2.159`. URL: `https://doi.org/10.1162/coli.2006.32.2.159`.

Schulte im Walde, Sabine, Helmut Schmid, Wiebke Wagner, Christian Hying, and Christian Scheible (2010). “A clustering approach to automatic verb classification incorporating selectional preferences: model, implementation, and user manual”. In:

Sener, Ozan and Vladlen Koltun (2018). “Multi-Task Learning as Multi-Objective Optimization”. In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 525–536. URL: `http://papers.nips.cc/paper/7334-multi-task-learning-as-multi-objective-optimization`.

Shi, Lei and Rada Mihalcea (2005). “Putting Pieces Together: Combining FrameNet, VerbNet and WordNet for Robust Semantic Parsing”. In: *Computational Linguistics and Intelligent Text Processing, 6th International Conference, CICLing 2005, Mexico City, Mexico, February 13-19, 2005, Proceedings*, pp. 100–111. DOI: `10.1007/978-3-540-30586-6\_9`. URL: `https://doi.org/10.1007/978-3-540-30586-6%5C_9`.

Shickel, Benjamin, Patrick Tighe, Azra Bihorac, and Parisa Rashidi (2017). “Deep EHR: A Survey of Recent Advances on Deep Learning Techniques for Electronic Health Record (EHR) Analysis”. In: *CoRR* abs/1706.03446. arXiv: 1706.03446. URL: http://arxiv.org/abs/1706.03446.

Sinclair, Stéfan and Geoffrey Rockwell (2012). “Introduction to Distant Reading Techniques with Voyant Tools, Multilingual Edition”. In: *Digital Humanities 2012, DH 2012, Conference Abstracts, University of Hamburg, Hamburg, Germany, July 16-22, 2012*, p. 26. URL: http://www.dh2012.uni-hamburg.de/conference/programme/abstracts/introduction-to-distant-reading-techniques-with-voyant-tools-multilingual-edition.1.html.

Smith, Larry et al. (2008). “Overview of BioCreative II gene mention recognition”. In: *Genome Biology* 9.2, S2. ISSN: 1474-760X. DOI: 10.1186/gb-2008-9-s2-s2.

Sniedovich, Moshe (2010). *Dynamic programming: foundations and principles.* CRC press.

Snoek, Jasper, Hugo Larochelle, and Ryan Prescott Adams (2012). “Practical Bayesian Optimization of Machine Learning Algorithms”. In: *CoRR* abs/1206.2944. arXiv: 1206.2944. URL: http://arxiv.org/abs/1206.2944.

Snyder, Benjamin and Martha Palmer (2004). “The English all-words task”. In: *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, SENSEVAL@ACL 2004, Barcelona, Spain, July 25-26, 2004*. URL: https://aclanthology.info/papers/W04-0811/w04-0811.

Springorum, Sylvia, Jason Utt, and Sabine Schulte im Walde (2013). “Regular Meaning Shifts in German Particle Verbs: A Case Study”. In: *Proceedings of the 10th International Conference on Computational Semantics, IWCS 2013, March 19-22, 2013, University of Potsdam, Potsdam, Germany*, pp. 228–239. URL: https://www.aclweb.org/anthology/W13-0120/.

Sudarikov, Roman, Ondrej Dusek, Martin Holub, Ondrej Bojar, and Vincent Kríz (2016). “Verb sense disambiguation in Machine Translation”. In: *Proceedings of the Sixth Workshop on Hybrid Approaches to Translation, HyTra@COLING, Osaka, Japan, December 11, 2016*, pp. 42–50. URL: https://aclanthology.info/papers/W16-4506/w16-4506.

Suk, Heung-Il and Dinggang Shen (2013). "Deep Learning-Based Feature Representation for AD/MCI Classification". In: *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2013 - 16th International Conference, Nagoya, Japan, September 22-26, 2013, Proceedings, Part II*, pp. 583–590. DOI: 10.1007/978-3-642-40763-5_72. URL: https://doi.org/10.1007/978-3-642-40763-5%5C_72.

Telljohann, Heike, Erhard W. Hinrichs, Sandra Kübler, Heike Zinsmeister, and Kathrin Beck (2012). *Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z)*. Seminar für Sprachwissenschaft. Wilhelmstr. 19, D-72074 Tübingen. URL: http://www.sfs.uni-tuebingen.de/fileadmin/static/ascl/resources/tuebadz-stylebook-1201.pdf.

Tixier, Antoine J.-P., Fragkiskos D. Malliaros, and Michalis Vazirgiannis (2016). "A Graph Degeneracy-based Approach to Keyword Extraction". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pp. 1860–1870. URL: https://www.aclweb.org/anthology/D16-1191/.

Turner, J. T., Adam Page, Tinoosh Mohsenin, and Tim Oates (2017). "Deep Belief Networks used on High Resolution Multichannel Electroencephalography Data for Seizure Detection". In: *CoRR* abs/1708.08430. arXiv: 1708.08430. URL: http://arxiv.org/abs/1708.08430.

Unterthiner, Thomas et al. (2014). "Deep learning as an opportunity in virtual screening". In: *Proceedings of the deep learning workshop at NIPS*. Vol. 27, pp. 1–9.

Usbeck, Ricardo et al. (2014). "AGDISTIS - Graph-Based Disambiguation of Named Entities Using Linked Data". In: *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, pp. 457–471. DOI: 10.1007/978-3-319-11964-9_29. URL: https://doi.org/10.1007/978-3-319-11964-9%5C_29.

Uslu, Tolga, Wahed Hemati, Alexander Mehler, and Daniel Baumartz (2017). "TextImager as a Generic Interface to R". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Software Demonstrations*, pp. 17–20. URL: https://www.aclweb.org/anthology/E17-3005/.

Uslu, Tolga and Alexander Mehler (2018). “PolyViz: a visualization system for a special kind of multipartite graphs”. In: *Proceedings of the IEEE VIS 2018*.

Uslu, Tolga, Alexander Mehler, and Daniel Baumartz (2019). “Computing Classifier-based Embeddings with the Help of text2ddc”. In: *Proceedings of the 20th International Conference on Computational Linguistics and Intelligent Text Processing, (CICLing 2019)*. CICLing 2019. La Rochelle, France.

Uslu, Tolga, Alexander Mehler, Daniel Baumartz, Alexander Henlein, and Wahed Hemati (2018). “fastSense: An Efficient Word Sense Disambiguation Classifier”. In: *Proceedings of the 11th edition of the Language Resources and Evaluation Conference, May 7 - 12*. LREC 2018. Miyazaki, Japan.

Uslu, Tolga, Alexander Mehler, and Dirk Meyer (2018). “LitViz: Visualizing Literary Data by Means of text2voronoi”. In: *Digital Humanities 2018, DH 2018, Book of Abstracts, El Colegio de México, UNAM, and RedHD, Mexico City, Mexico, June 26-29, 2018*, pp. 308–311. URL: `https://dh2018.adho.org/en/litviz-visualizing-literary-data-by-means-of-text2voronoi/`.

Uslu, Tolga, Lisa Miebach, et al. (2018). “Automatic Classification in Memory Clinic Patients and in Depressive Patients”. In: *Proceedings of Resources and ProcessIng of linguistic, para-linguistic and extra-linguistic Data from people with various forms of cognitive/psychiatric impairments*. RaPID. Miyazaki, Japan.

Veronis, Jean (2001). “Sense tagging: does It make sense?” In: *Corpus Linguistics’2001 Conference*.

Vickrey, David, Luke Biewald, Marc Teyssier, and Daphne Koller (2005). “Word-Sense Disambiguation for Machine Translation”. In: *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*, pp. 771–778. URL: `http://aclweb.org/anthology/H/H05/H05-1097.pdf`.

Vossen, Piek (1998). “Introduction to EuroWordNet”. In: *Computers and the Humanities* 32.2-3, pp. 73–89. DOI: `10.1023/A:1001175424222`. URL: `https://doi.org/10.1023/A:1001175424222`.

Wagner, Wiebke, Helmut Schmid, and S Schulte Im Walde (2009). “Verb sense disambiguation using a predicate-argument-clustering model”. In: *Proceedings* *of the CogSci Workshop on Distributional Semantics beyond Concrete Concepts.* Citeseer, pp. 23–28.

Wang, Alex et al. (2018). “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding”. In: *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pp. 353–355. URL: `https://www.aclweb.org/anthology/W18-5446/`.

Wei, Chih-Hsuan et al. (2016). “Assessing the state of the art in biomedical relation extraction: overview of the BioCreative V chemical-disease relation (CDR) task”. In: *Database* 2016. DOI: `10.1093/database/baw032`. URL: `https://doi.org/10.1093/database/baw032`.

Weisstein, Eric W (2002). “Sigmoid function”. In:

Weller, Marion, Sabine Schulte Im Walde, and Alexander Fraser (2014). “Using noun class information to model selectional preferences for translating prepositions in smt”. In: *Proceedings of the 11th Conference of the Association for Machine Translation in the Americas*, pp. 275–287.

Wiktionary. `https://www.wiktionary.org/`. Accessed: 2019-09-23.

Wistuba, Martin, Nicolas Schilling, and Lars Schmidt-Thieme (2015). “Learning hyperparameter optimization initializations”. In: *2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, Campus des Cordeliers, Paris, France, October 19-21, 2015*, pp. 1–10. DOI: `10.1109/DSAA.2015.7344817`. URL: `https://doi.org/10.1109/DSAA.2015.7344817`.

Witten, Ian H., Eibe Frank, and Mark A. Hall (2011). *Data mining: practical machine learning tools and techniques, 3rd Edition.* Morgan Kaufmann, Elsevier. ISBN: 9780123748560. URL: `http://www.worldcat.org/oclc/262433473`.

Wu, Xindong, Xingquan Zhu, Gong-Qing Wu, and Wei Ding (2014). “Data Mining with Big Data”. In: *IEEE Trans. Knowl. Data Eng.* 26.1, pp. 97–107. DOI: `10.1109/TKDE.2013.109`. URL: `https://doi.org/10.1109/TKDE.2013.109`.

Wu, Zhibiao and Martha Stone Palmer (1994). “Verb Semantics and Lexical Selection”. In: *32nd Annual Meeting of the Association for Computational Linguistics, 27-30 June 1994, New Mexico State University, Las Cruces, New Mexico, USA, Proceedings.* Pp. 133–138. URL: `http://aclweb.org/anthology/P/P94/P94-1019.pdf`.

Xu, Kelvin et al. (2015). "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". In: *CoRR* abs/1502.03044. arXiv: 1502.03044. URL: http://arxiv.org/abs/1502.03044.

Xu, Shuo, Xin An, Lijun Zhu, Yunliang Zhang, and Haodong Zhang (2015). "A CRF-based system for recognizing chemical entity mentions (CEMs) in biomedical literature". In: *J. Cheminformatics* 7.S-1, S11. DOI: 10.1186/1758-2946-7-S1-S11. URL: https://doi.org/10.1186/1758-2946-7-S1-S11.

Yang, Zhilin et al. (2019). "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: *CoRR* abs/1906.08237. arXiv: 1906.08237. URL: http://arxiv.org/abs/1906.08237.

Yang, Zichao, Xiaodong He, Jianfeng Gao, Li Deng, and Alexander J. Smola (2015). "Stacked Attention Networks for Image Question Answering". In: *CoRR* abs/1511.02274. arXiv: 1511.02274. URL: http://arxiv.org/abs/1511.02274.

Yeh, Alexander S., Alexander A. Morgan, Marc E. Colosimo, and Lynette Hirschman (2005). "BioCreAtIvE Task 1A: gene mention finding evaluation". In: *BMC Bioinformatics* 6.S-1. DOI: 10.1186/1471-2105-6-S1-S2. URL: https://doi.org/10.1186/1471-2105-6-S1-S2.

Zhang, Lu, Jianjun Tan, Dan Han, and Hao Zhu (2017). "From machine learning to deep learning: progress in machine intelligence for rational drug discovery". In: *Drug Discovery Today* 22.11, pp. 1680–1685. ISSN: 1359-6446.

Zhao, Yilu and Lianghua He (2014). "Deep Learning in the EEG Diagnosis of Alzheimer's Disease". In: *Computer Vision - ACCV 2014 Workshops - Singapore, Singapore, November 1-2, 2014, Revised Selected Papers, Part I*, pp. 340–353. DOI: 10.1007/978-3-319-16628-5_25. URL: https://doi.org/10.1007/978-3-319-16628-5%5C_25.

Zhong, Zhi and Hwee Tou Ng (2012). "Word Sense Disambiguation Improves Information Retrieval". In: *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, pp. 273–282. URL: http://www.aclweb.org/anthology/P12-1029.

A. Appendix

A.1. Zusammenfassung

Ein Ziel dieser Arbeit war die Entwicklung eines Frameworks für die Homogenisierung der NLP-Landschaft, das es verschiedenen spezialisierten NLP-Werkzeugen ermöglicht, interoperabel zu werden, nämlich TextImager. Dabei führt TextImager bestehende Systeme und Entwicklungsumgebungen modular zusammen, um die integrierten Module untereinander austauschbar zu machen und Spezialisierungsvorteile unter den Modulen nutzbar zu machen. Um die immer größer werdenden Datenmengen verarbeiten zu können, ist das Framework in der Lage, cluster-basiert zu laufen und dabei sowohl horizontal als auch vertikal skalierbar zu sein.

Die Tabelle A.1 zeigt die Verarbeitungszeit mit dem TextImager für den Korpus, der im Abschnitt 3.1 vorgestellt wurde. Zu erwähnen ist, dass die Architektur des TextImagers die Verarbeitung um das 7-fache beschleunigt hat. Dieser Faktor kann weiter erhöht werden, da die Infrastruktur es nun ermöglicht, zusätzliche Instanzen dynamisch zu starten.

TextImager wird bereits von vielen Arbeiten als Vorverarbeitungspipeline, als Feature-Generator und als Programmierschnittstelle eingesetzt (Mehler, Abrami, et al. 2018; Baumartz, Uslu, and Mehler 2018; Hemati and Mehler 2019a; Hemati and Mehler 2019b; Kett et al. 2018; Abrami, Mehler, Lücking, et al. 2019; Abrami, Mehler, and Spiekermann 2019; Rutherford, Hemati, and Mehler 2018; Hunziker et al. 2019; Uslu, Mehler, and Baumartz 2019; Uslu and Mehler 2018; Uslu, Mehler, and Meyer 2018; Mehler, Hemati, Uslu, et al. 2018; Uslu, Hemati, et al. 2017; Hemati, Uslu, and Mehler 2017). Das Projekt ist auf GitHub frei zugänglich und wird von vielen Beitragenden bereits weiterentwickelt. Eine der zukünftigen

Table A.1.: Statistiken über die zeitliche und räumliche Komplexität des verarbeiteten Korpus, die in Abschnitt 3.1 vorgestellt wurde. Im Vergleich zur nicht verteilten Version ist Textimager mehr als 7 mal schneller geworden.

	Step	**Days**	**GB**
1	Tokenize	0.32	82.48
2	Lemmatization	0.84	97.70
3	POS Tagging	0.37	94.42
4	Named Entity Recognition	0.47	9.06
5	Dependency Parsing	1.85	225.46
6	Time Recognition	0.99	11.95
7	Sentiment Analysis	0.76	7.13
8	Semantic Role Labeling	0.9	8.09
9	Wikification	1.24	26.40
10	Coreference	2.58	6.74
	$\sum$	10.32	569.43

Arbeiten wird darin bestehen, einen Prozess zur Automatisierung der Integration neuer Tools[1] zu entwickeln, so dass TextImager immer auf dem neuesten Stand der Forschung ist.

Im Zuge der Entwicklung der Computerlinguistik hat sich ein rudimentäres Prozessmodell als wissenschaftliche Disziplin etabliert, das praktizierte Sequenzregularitäten von NLP-Werkzeugen in eine Pipeline Reihenfolge definiert. Der Vorteil einer solchen NLP-Pipeline besteht darin, dass komplexere Aufgaben in kleinere Aufgaben unterteilt werden können, für die spezialisierte Werkzeuge entwickelt werden können. TextImager implementiert eine solche Pipeline-Struktur. Durch die integrierten Werkzeuge und die homogenisierten I/O-Datenströme des TextImagers ist es zudem möglich, die eingebauten Werkzeuge auf zwei Dimensionen zu kombinieren: (1) die horizontale Dimension, um eine NLP aufgabenspezifische Verbesserung zu erreichen (2) die orthogonale Dimension, um CL Modelle implementieren zu können, die auf mehrere Ebenen der Sprache aufbauen und somit auf eine Kombination aus unterschiedlichen NLP Werkzeugen angewiesen sind.

[1] `https://paperswithcode.com/area/natural-language-processing`

Dies bringt allerdings auch Nachteile mitsich. Fehler in der Verarbeitungskette werden an nachfolgende Werkzeuge weitergegeben. Um dies zu vermeiden, müsste die gesamte Verarbeitung als ein einziges Optimierungs-/Lernproblem formuliert werden (Mehler, Hemati, Gleim, et al. 2018) . Einzelne Schritte können sich so besser gegenseitig beeinflussen. Es gibt bereits Ansätze, die diese Art der Single-Optimierung durchführen, nämlich Multi-Task-Learning (MLT). Als zukünftige Arbeit muss die explizite Pipeline- und aufgabenorientierte sequentielle NLP-Verarbeitung, wie sie derzeit vom TextImager implementiert wird, mit einem dynamischen MLT prozess kombiniert werden. Eine mögliche Lösung wäre ein hierarchisches Modell für MLT (Sanh, Wolf, and Ruder 2019). Das Modell wird hierarchisch trainiert, um einen induktiven Bias einzuführen, indem es eine Reihe von Low-Level-Aufgaben (z.B. Lemmatisierung und POS-Tagging) in den unteren Schichten des Modells und komplexere Aufgaben (z.B. SRL, NER, etc.) in den oberen Schichten des Modells trainiert. Diese Ansätze werden in zukünftigen Arbeiten angegangen.

In dieser Arbeit wurde des weiteren ein großer Schritt in die semantische Analyse von Verben für das Deutsche gemacht. Es wurde ein state-of-the-art Verb-Sense-Disambiguierungssystem für Deutsch entwickelt. Um dies zu erreichen, wurde das größte verbsinnannotierte deutsche Korpus erstellt, das für die Erstellung von überwachten Machine Learning Verfahren benötigt wird. Das Korpus umfasst 80 % der deutschen Verb-Token, gemessen an COW. Dabei decken 20 % der GermaNet Verb-Lemmata 80 % der Verb-Token. Einige verbleibende Verben aus GermaNet sind jedoch noch nicht abgedeckt. Darüber hinaus gibt es Verben, die nicht im GermaNet aufgeführt sind. Eine fortlaufende manuelle Annotation für die Erstellung von Trainingsdaten für die noch nicht abgedeckten Verben wird auf dauer zu aufwending. Um dies entgegenzuwirken, müssen Methoden der distributiven und deklarativen Semantik im Zusammenhang mit VSD erforscht werden. Die distibutive Semantik ist ein Forschungsgebiet, das Theorien und Methoden zur Quantifizierung und Kategorisierung semantischer Ähnlichkeiten zwischen linguistischen Elementen erforscht.Es kann in der Distributionshypothese zusammengefasst werden: Linguistische Elemente mit ähnlichen Verteilungen haben ähnliche Bedeutungen (Harris 1954; Dahl 2016). Verteilungsmodelle repräsentieren ein Wort durch den Kontext, in denen es vorkommt, wobei moderne Ansätze Vektorraummodelle oder Wordembeddings implementieren (Mikolov, Chen, et al. 2013; Blei, A. Y. Ng, and Jordan 2003; Deerwester et al. 1990). In diesen Modellen werden Wörter als Punkte in einem hochdimensionalen Raum dargestellt, in dem ähnliche Wörter tendenziell näher beieinander liegen. Der Nachteil ist, dass pro syntaktischem Wort ein Vektor erzeugt wird, was bedeutet, dass die Sinnebene nicht erfasst wird. Aktuelle Modelle, wie z.B. `BERT` (Devlin, M. Chang, et al. 2019), `XLNet.` (Zhilin Yang et al. 2019) und `ELMo`, induzieren kontextsensitive Wordembeddings. Die Idee ist es, implizit Sinnembeddings zu erzeugen, die durch verschiedene Vektoren für das gleiche syntaktische Wort in verschiedenen Kontexten repräsentiert werden. In zukünftigen Arbeiten wird diese Art der impliziten Semantik verwendet, um Sinneunterschiede für Verben zu erhalten, für die keine Trainingsdaten verfügbar sind und daher kein supervised ML trainiert werden kann.

Eine weitere Möglichkeit, eine höhere Abdeckung der Verb-Semantik zu erreichen, ist die Verwendung der deklarativer Semantik. Im Falle der deutschen Verben sind Partikelverben besonders nützlich, da sie hochproduktiv sind (Springorum, Utt, and Schulte im Walde 2013). Die größte Herausforderung dieser Verben ist die der Kompositionalität: Kann die Bedeutung eines solchen Verbs durch das Partikel oder Präfix und dem Rest vorhergesagt werden? Die Frage bleibt offen, ob ein algebraisches Modell entwickelt werden kann, um die Bedeutung zu approximieren, basierend auf der Kombination der Bedeutung des Basisverbes und des Partikels (Bott and Schulte im Walde 2018; Köper et al. 2016). Die Beantwortung dieser Frage wird in einer zukünftigen Arbeit thematisiert.

Es gibt bekanntlich einen Zusammenhang zwischen Subkategorisierungsrahmen und Verb-Sinnen. Levin (1993) Klassifizierung basiert auf der Hypothese, dass das syntaktische Verhalten eines Verbs und seine Bedeutung stark miteinander verknüpft sind. Ähnlich wie die Levin (1993) Klassen können deutsche Verben in semantische Verbenklassen unterteilt werden, um unbekannte Verben semantisch zuzuordnen (Schulte im Walde 2006; Scheible et al. 2013). In Zukunft werden wir die neuartigen kontextsensitiven Embeddings in Kombination mit Subkategorisierungsrahmen und Prädikat-Argument-Struktur von Verben (W. Wagner, Schmid, and S Schulte Im Walde 2009; Schulte im Walde et al. 2010) kombinieren, um selektive Restriktionen (Weller, Sabine Schulte Im Walde, and Fraser 2014; Mu, Hartshorne, and O'Donnell 2017) für cluster-basierte Sinn-Disambiguierung und Sinn-Induktion zu extrahieren.

A.2. Verbs

A.2.1. Skinners Law Evaluation

Wikipedia Title	S1	S2
Großer_Sprung_nach_vorn	1	4
Gutmensch	4	0
Narzissmus	4	2
Schlacht_auf_den_Katalaunischen_Feldern	0	2
Instant-Runoff-Voting	2	0
Kniefall_von_Warschau	2	0
Herakleios	0	2

Table A.2.: Sense distribution for *übertreiben*

Wikipedia Title	S1	S2	S3	S4
Tunesien	1	19	0	0
Holzminden	1	34	0	0
1910er	0	31	0	0
Welfen	0	3	1	0
Friedensbewegung	0	14	0	0

Table A.3.: Sense distribution for *gründen*

Wikipedia Title	S1	S2
Fanservice	3	0
Steirische_Harmonika	0	21
FC_Arsenal	3	1
Freie_Software	3	0
Tandy_TRS-80_Model_1	0	2

Table A.4.: Sense distribution for *einbauen*

Wikipedia Title	S1	S2
Eiger-Nordwand	3	0
Steve_Winwood	0	2
Amerikanischer_Schwarzbär	4	0
Peter_und_der_Wolf	2	0
The_Beach_Boys	0	4
Totenkopfschwärmer	3	0
Grünspecht	2	0
Eiffelturm	1	1
World_Trade_Center	2	0
Titanic_(1997)	2	0

Table A.5.: Sense distribution for *klettern*

Wikipedia Title	S1	S2	S3	S4	S5
Else_Lasker-Schüler	0	0	0	11	0
Heinrich_VII._(HRR)	0	0	0	11	0
HTML-Editor	0	2	0	0	0
Zerspanbarkeit	0	6	0	0	0
Interactive_System_Productivity_Facility	0	4	0	0	0

Table A.6.: Sense distribution for *bearbeiten*

Wikipedia Title	S1	S2
Victoria_von_Großbritannien_und_Irland_(1840?1901)	1	2
Mayhem	0	3
Karate	2	0
Go_(Spiel)	2	0
Unabhängige_Sozialdemokratische_Partei_Deutschlands	0	3
Kneipe_(Studentenverbindung)	3	0

Table A.7.: Sense distribution for *begrüßen*

Wikipedia Title	S1	S2	S3	S4
Gefangenendilemma	28	0	0	0
1984_(Roman)	7	0	0	0
Figuren_in_Tolkiens_Welt	5	1	0	0
Marcus_Antonius	1	3	0	0
Johannes_Bückler	2	2	0	0
Anne_Frank	3	1	0	0
Doppelstern	0	0	1	2

Table A.8.: Sense distribution for *verraten*

Wikipedia Title	S1	S2	S3	S4
Figuren_in_Tolkiens_Welt	0	12	0	1
Adolf_Hitler	0	9	0	0
Verschwörungstheorie	0	1	1	1
Buffy_–_Im_Bann_der_Dämonen	0	7	1	1

Table A.9.: Sense distribution for *erfahren*

Wikipedia Title	S1	S2	S3	S4
Meteorit	1	0	9	0
Tower_of_London	2	13	0	0
Warschau	1	14	0	2
Zirkon	0	0	8	6

Table A.10.: Sense distribution for *stammen*

Wikipedia Title	S1	S2	S3
Steirische_Harmonika	0	25	0
Die_Siedler_von_Catan	0	16	0
Arche_Noah	0	12	0

Table A.11.: Sense distribution for *bauen*

A.2.2. GermaNet Sense Mappings to Super Senses

The table shows the merged senses and the respective decision criteria:

- 🟦 Senses not distinguishable
- 🟥 Circular Senses
- 🟩 Senses/distinctions are missing
- 🟧 Obsolete or dialectical meanings
- ⬛ Methaphor

LexIds	Map To	Lemma	C.
78225	76100	ablehnen	🟦
79173	78279	ablehnen	🟥
78263	78279	ablehnen	🟥
83482	83480	abschließen	🟦
144567	144566	abspielen	🟦
75468	75463	abstimmen	🟦
77711	74980	agieren	🟥
75668	74980	agieren	🟥
79573	74040	anbieten	🟥
75755	74040	anbieten	🟥
76330	83407	anfangen	🟦
83272	78924	anführen	🟦
79800	79740	angehen	🟦
79517	78181	anlocken	⬛
76490	74114	annehmen	🟦
75163	74114	annehmen	🟦
77336	77249	annehmen	🟦
79535	78077	anordnen	🟥
83780	75422	anpassen	⬛
82446	82402	ansehen	🟦
82445	82402	ansehen	🟦
75659	144803	ansiedeln	⬛
80564	76263	anwenden	🟦
77735	76263	anwenden	🟦
144832	75543	anzeigen	🟥
77955	77709	arbeiten	🟦
79738	79207	attackieren	🟦
75850	83145	aufbauen	🟦
78434	85400	aufdecken	🟦
79554	76194	auferlegen	🟦
83470	79874	aufgeben	🟦
83497	85392	aufheben	🟦
83504	73727	aufhören	🟦
77580	77882	aufklären	🟥
78832	77882	aufklären	🟦
82438	77430	aufpassen	🟦
74690	74898	aufregen	🟦
144916	74898	aufregen	🟥
82315	77888	aufspüren	🟥
80824	80818	aufstellen	🟦
83259	78652	aufstellen	🟦
82739	81866	auftauchen	🟦
77554	81866	auftauchen	⬛
85538	75835	aufteilen	🟦
75671	75667	auftreten	🟥
83814	82740	auftreten	🟥
82725	74394	aufweisen	🟦
84888	84886	ausbauen	🟦
84887	84886	ausbauen	🟦
83156	78555	ausdenken	🟦
77474	74521	aushalten	🟦
77462	74521	aushalten	🟦
83426	83190	auslösen	🟦
145113	76111	ausschalten	🟦
78829	78613	aussprechen	🟦
145187	84768	austauschen	🟥
145195	83519	ausweichen	⬛
73494	73491	auszeichnen	🟦

LexIds	Map To	Lemma	C.
82930	82896	bauen	🟥
77382	79034	beanspruchen	🟦
74672	74678	bedauern	🟥
82700	80406	bedecken	🟦
74853	73640	beeindrucken	🟦
84840	78080	beeinflussen	🟥
84870	79663	beeinträchtigen	🟦
145236	80003	befestigen	🟦
76443	76256	befriedigen	🟦
82286	77712	begegnen	🟦
82320	75176	begegnen	⬛
83406	145239	beginnen	🟥
81169	75945	begleiten	🟦
109526	79013	begründen	🟥
79021	78337	beharren	🟥
79766	77478	behaupten	🟦
109404	79094	bekräftigen	🟦
145263	79803	bekämpfen	⬛
76219	73964	belohnen	🟦
77420	75553	bemühen	🟦
78041	75368	benennen	🟦
85957	76270	benutzen	🟦
77750	78343	berücksichtigen	🟥
74239	75567	beschaffen	⬛
76509	77950	beschäftigen	🟦
109437	79935	besetzen	🟦
109435	79935	besetzen	🟦
75566	75031	besorgen	🟦
145311	78029	bestimmen	🟦
109454	75372	bestimmen	🟦
78328	78324	bestätigen	🟦
79082	78324	bestätigen	🟦
77483	75262	besuchen	🟦
141358	76528	betreffen	🟦
75802	75324	betreiben	🟦
80757	80753	bewegen	🟦
78441	82734	beweisen	🟦
78598	82734	beweisen	🟦
109317	73988	bezahlen	🟦
109316	73988	bezahlen	🟦
77734	79049	beziehen	🟦
76533	79049	beziehen	🟦
74039	75746	bieten	🟥
83873	75746	bieten	🟥
75779	75746	bieten	🟦
79585	77993	billigen	🟦
79164	75057	binden	🟦
82299	82303	blicken	🟦
85323	76113	blockieren	🟦
85315	76113	blockieren	🟦
77378	85724	brauchen	🟦
85727	85724	brauchen	🟦
84250	83725	brechen	🟦
76300	83725	brechen	🟦
81248	73921	bringen	🟦
78032	73765	charakterisieren	🟦
78975	78552	darlegen	🟦
73766	73304	darstellen	🟦
78976	78551	darstellen	🟦
78954	78551	darstellen	🟦
109332	78593	demonstrieren	🟦
77708	77789	denken	🟥
83258	78596	dokumentieren	🟦
82808	82055	drehen	🟦
81914	82055	drehen	🟦
83349	79622	drucken	🟦
81188	80691	drängen	🟦
75872	75023	durchführen	🟦
75866	75023	durchführen	🟦
79887	76367	durchsetzen	🟦
76240	73457	eignen	🟥
78345	73551	einbeziehen	🟦

LexIds	Map To	Lemma	C.
77752	73551	einbeziehen	🟦
77963	75164	eingehen	🟦
77373	77361	einrichten	🟦
85175	74094	einräumen	🟦
76493	76492	einsetzen	🟦
77362	75462	einstellen	🟦
144378	74209	empfangen	🟦
82487	74485	empfinden	🟦
83548	83535	enden	🟦
82306	77588	entdecken	🟦
83174	78984	entfalten	🟦
78044	76437	entscheiden	🟦
76222	73963	entschädigen	🟦
76442	73437	entsprechen	🟦
83158	78543	entwerfen	🟦
83036	78535	entwickeln	🟥
84008	83834	entwickeln	🟦
83882	83834	entwickeln	🟦
109986	74318	erarbeiten	🟦
74571	74547	erfreuen	🟦
73413	76454	erfüllen	🟦
78581	73745	ergeben	🟦
74434	73745	ergeben	🟦
84937	77818	ergänzen	🟦
83883	78308	erheben	🟦
74724	77109	erholen	🟦
84039	84038	erhöhen	🟦
82264	82262	erkennen	🟦
78970	78895	erklären	🟦
89997	74211	erlangen	⬛
76088	78311	erlauben	🟦
77545	75260	erleben	🟥
77541	75260	erleben	🟦
79714	74515	erleiden	🟦
74657	74515	erleiden	🟦
77886	82321	ermitteln	🟦
82764	76087	ermöglichen	🟦
79193	79923	erobern	🟦
110251	78567	erschließen	🟦
100797	74609	erschrecken	🟦
77454	74518	ertragen	🟦
77331	77396	erwarten	🟦
74237	74322	erwerben	🟦
78960	78959	erzählen	🟦
83450	75849	eröffnen	🟦
144397	83148	etablieren	🟦
81239	81559	fahren	🟥
81634	81559	fahren	🟥
87060	73571	fehlen	🟦
87224	84801	festigen	🟦
78740	75095	festlegen	🟦
82261	77584	feststellen	🟦
77892	77584	feststellen	🟦
82307	77891	finden	🟦
81546	81620	fliegen	🟥
141265	81350	fliegen	🟦
79030	77376	fordern	🟦
112657	78321	freigeben	🟦
74620	74602	fürchten	🟦
75118	73801	geben	🟦
81724	81356	gehen	🟥
130725	73519	gehen	🟦
73387	73375	geschehen	🟦
78313	76090	gestatten	🟦
78313	76090	gestatten	🟦
77245	77229	glauben	🟦
82690	82239	glänzen	🟦
78194	73600	halten	🟥
77745	73600	halten	⬛
77593	73600	halten	🟦
77652	76286	halten	🟥
74370	73671	halten	🟦

LexIds	Map To	Lemma	C.
73856	73815	handeln	🟦
83800	77800	heben	🟦
83793	84749	heilen	🟦
82323	77583	herausfinden	🟦
78781	78775	hervorheben	🟦
79668	76127	hindern	🟥
75265	75216	hingehen	🟦
77991	74519	hinnehmen	🟦
82728	78787	hinweisen	🟦
82450	82447	hören	🟥
77481	82447	hören	🟦
74870	78174	inspirieren	🟦
77244	77241	kennen	🟥
77242	77241	kennen	🟦
74728	82603	klagen	🟦
80318	80310	klopfen	🟦
78529	78522	klären	🟦
84083	73789	kommen	🟦
77713	79643	konfrontieren	🟦
78129	75814	kontrollieren	🟦
83243	85706	kopieren	🟦
82863	85706	kopieren	🟦
141069	79789	kämpfen	🟦
81843	81834	landen	🟧
81449	81357	laufen	🟦
83806	73401	laufen	🟦
109367	76423	lauten	🟦
73265	76674	leben	🟥
83944	74723	legen	🟦
86971	75707	lehren	🟦
79287	77523	lesen	🟦
82677	82207	leuchten	🟦
79516	74501	locken	🟦
78179	74501	locken	⬛
140156	77196	locken	🟦
78509	76298	lösen	🟦
78426	76298	lösen	🟦
77579	76298	lösen	🟦
83092	83110	malen	🟦
86797	86794	melden	🟦
86796	86794	melden	🟦
110714	80694	mischen	🟦
77600	82281	mitbekommen	🟦
75241	75250	mitmachen	🟦
74249	81171	mitnehmen	🟦
140604	80058	montieren	🟦
74626	73584	mögen	🟦
77590	78574	nehmen	🟥
85914	74109	nehmen	🟥
80339	74109	nehmen	🟦
73793	73792	neigen	🟦
79449	76412	nennen	🟦
74900	74688	nerven	🟦
78357	75851	organisieren	🟦
75569	74255	organisieren	🟦
141981	80361	packen	⬛
112508	112507	probieren	🟦
82766	78517	produzieren	🟦
142056	75742	promovieren	🟦
142072	75735	qualifizieren	🟦
86970	85872	rauchen	🟦
110711	75589	regeln	🟦
141611	82907	rekonstruieren	🟦
85174	75822	räumen	🟦
82749	78518	schaffen	🟥
82781	78518	schaffen	🟦
129735	79300	schimpfen	🟥
85814	129775	schmecken	🟦
87037	74801	schreien	🟦
141668	84827	schwächen	🟦
79748	76018	schützen	🟦
74386	74371	sparen	🟦

LexIds	Map To	Lemma	C.
83017	74363	speichern	🟥
79286	78950	sprechen	🟦
81463	80765	springen	🟦
82488	74489	spüren	🟦
80952	80958	stammen	🟥
80957	80958	stammen	🟦
83441	75871	starten	🟦
130045	74497	staunen	🟦
79999	80440	stecken	🟦
80446	80440	stecken	🟦
89378	89380	stecken	⬛
84903	77806	steigern	🟥
80844	80813	stellen	🟦
82666	76837	stinken	🟦
81861	83502	stoppen	🟦
81201	81093	stoßen	🟦
82679	82208	strahlen	🟦
145181	83179	strahlen	🟦
141822	79772	streiten	🟦
83764	84804	stärken	🟦
89400	79649	stören	🟦
73751	75953	stützen	🟦
81986	75683	tanzen	🟦
75197	77276	trauen	🟥
75273	75175	treffen	🟦
89423	89422	treten	🟦
130357	82360	umsehen	🟦
83973	75159	unterbringen	🟦
130381	75863	unternehmen	🟦
86282	79915	unterwerfen	🟦
78656	76196	urteilen	🟥
78729	75108	verabschieden	🟦
130400	85386	verbergen	🟦
84688	83789	verbessern	🟦
82970	85720	verbrauchen	🟦
110875	74215	verbuchen	🟦

LexIds	Map To	Lemma	C.
81361	77414	verfolgen	🟦
79560	74337	verfügen	🟥
78031	74337	verfügen	🟥
74405	74337	verfügen	🟦
130457	75012	vergewaltigen	🟦
73296	73645	verhalten	🟦
130471	78674	verhandeln	🟦
78804	75070	verheiraten	🟦
84852	84067	verkürzen	🟦
75925	77318	verlangen	🟦
83938	74423	verlieren	🟦
84003	84923	verlängern	🟦
112505	75571	vermitteln	🟦
111004	76022	vernachlässigen	🟦
84659	79919	vernichten	🟦
84262	79919	vernichten	🟦
131539	78078	verordnen	🟦
112413	75223	verpassen	🟥
112409	75223	verpassen	🟥
79171	75099	verpflichten	🟦
78744	78812	verraten	🟦
81224	81074	verschieben	🟦
75762	79159	versprechen	🟦
89447	78850	verständigen	🟦
79792	79744	verteidigen	🟦
76011	79011	verteidigen	🟦
78361	85576	verteilen	🟦
132277	75434	vertragen	🟥
82963	76271	verwenden	🟦
77400	79042	vorbehalten	🟦
132404	79808	vordringen	🟦
112510	82326	vorfinden	🟦
78653	75694	vorgeben	🟦
77366	77365	vorsehen	🟦
78570	77596	vorstellen	🟥
76414	76413	vorstellen	🟦

LexIds	Map To	Lemma	C.
132715	78967	vortragen	🟦
82721	73940	vorweisen	🟦
109707	109708	wachen	🟦
84007	76735	wachsen	🟦
83859	84024	wachsen	🟥
80590	84998	wachsen	🟦
77275	75194	wagen	🟦
83556	73391	wandeln	🟦
78918	78913	warnen	🟦
89494	73656	warten	🟦
76055	73656	warten	🟦
73824	73823	wechseln	🟥
89501	84143	wechseln	🟦
85060	74157	wegnehmen	🟥
132876	82251	wehen	🟥
112234	79746	wehren	🟦
133237	79595	weiterleiten	🟦
133293	133286	wenden	🟦
109333	79199	werben	🟦
84147	77510	wiederholen	🟦
113289	82738	wiederspiegeln	🟦
73643	73637	wirken	🟦
83180	73637	wirken	🟦
73329	73312	wohnen	🟦
74008	73967	zahlen	🟥
89629	83077	zeichnen	🟦
83101	83077	zeichnen	🟦
73628	78592	zeigen	🟦
113100	78428	zerlegen	🟦
81203	81075	ziehen	🟦
79069	78227	zugeben	🟦
78315	76091	zulassen	🟦
139606	78532	zurückführen	🟦
74848	73307	zusammenhängen	🟦
75845	74281	zusammenstellen	🟦
78533	78004	zuschreiben	🟦
139871	77996	zustimmen	🟦
84160	78231	ändern	🟥
78608	78742	äußern	🟥
83970	85366	öffnen	🟦
83965	85376	öffnen	🟦
73739	73831	überlassen	🟦
74111	74110	übernehmen	🟥
130392	73677	übersehen	🟦
82436	76079	überwachen	🟦
139979	76299	überwinden	🟦