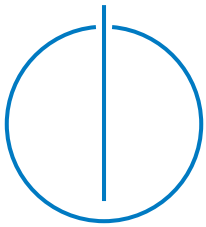
Fakultät für Informatik
Der Technischen Universität München

Bachelor's Thesis in Informatics

Sentence Boundary Detection in German Legal Documents

Sebastian Moser



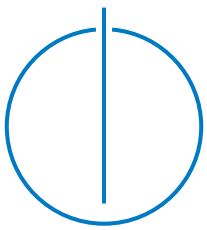
Fakultät für Informatik
Der Technischen Universität München

Bachelor's Thesis in Informatics

Sentence Boundary Detection in deutschsprachigen Rechtstexten

Sentence Boundary Detection in German Legal Documents

Erstbetreuer:	Prof. Dr. Florian Matthes
Zweitbetreuer:	M.Sc. Ingo Glaser
Tag der Einreichung:	16.08.2019



Erklärung

Ich versichere, dass ich diese Bachelor's Thesis selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

I assure the single handed composition of this bachelor's thesis only supported by declared resources.

München, den 13. August 2019

Sebastian Moser

Zusammenfassung

Satzsegmentierung oder Sentence Boundary Detection in deutschen Rechtstexten ist ein Aufgabe mit dem standartisierte Textverarbeitungssysteme wenig bis gar nicht umgehen können, da diese durch darin enthaltenen komplexere Strukturen wie Listen, Paragraphenstrukturen und Zitationen teilweise überfordert werden. In dieser Arbeit werden zum einen die Leistungen von diesen Systemen evaluiert, zum anderen werden Methoden direkt an die Rechtsdomäne angepasst.

Die erstellten Neuronalen Netzwerke und Conditional Random Fields können signifikant höhere Leistungen aufweisen als die getesteten, bereits bestehenden Systeme OpenNLP und NLTK. Damit steht diese Bachelorarbeit im Widerspruch zur häufigen Annahme, dass die Satzsegmentierung bereits gelöst sei. Angepasste regelbasierte Methoden werden ebenfalls angewendet, weisen aber ähnliche Leistungsschwächen auf.

Außerdem wurde ein neuer Dokumentenkorpus mit über 50.000 Sätzen erstellt, der auch in Zukunft dazu dienen soll SBD im Rechtsbereich weiter zu erforschen. Der Datensatz besteht hauptsächlich aus Gesetzen und Urteilen mit einem kleineren Teil bestehend aus AGBs von Onlineshops, Datenschutzerklärungen, Wikipediaartikeln und Gesetzen im XML-Format.

Abstract

Sentence Boundary Detection in German legal documents poses a challenge for standardized NLP-systems, which can only be dealt with partly or not at all because of their complex structures such as lists, paragraph structures and citations. On the one hand, the performance of those systems is evaluated. On the other hand, methods are directly tailored to the legal domain.

The created Neural Networks and Conditional Random Fields are able to outperform the tested, existing systems OpenNLP and NLTK by a great margin. Thus this Bachelor's Thesis contradicts the common assumption that sentence segmentation is already solved. Tailored rule-based systems are also applied but show the same performance weaknesses.

Additionally, a new dataset with over 50,000 sentences is created, which can be used for further research on SBD in the German legal domain. The document corpus consists mainly of laws and judgments with a small part made up of terms of service from online shops, privacy policies, Wikipedia articles and laws in XML-format.

Contents

Abbreviations

BGB	Bürgerliches Gesetzbuch, German Civil Code
CDC	Conan Doyal Corpus
CRF	Conditional Random Fields
GENIA	Corpus of abstracts of papers from the biological domain
GG	Grundgesetz
GRU	Gated Recurrent Unit
GUI	Graphical User Interface
HMM	Hidden Markov Models
LSTM	Long Short-Term Memory
ME	Maximum Entropy Models
MUC'7	Corpus from 7th Message Understanding Conference
NLP	Natural Language Processing
NN	Neural Network
POS	Part-of-Speech
RNN	Recurrent Neural Network
SBD	Sentence Boundary Detection
SGB	Sozialgesetzbuch, Social Security Code
StGB	Strafgesetzbuch, German Criminal Code
SVM	Support Vector Machine
WSJ	Wall Street Journal Corpus

List of Figures

List of Tables

1. Introduction

Most of the work done in the legal domain is related to some text or document. Argumentations, responses to judgments, reading/writing contracts, etc. require a thorough legal knowledge, and most tasks are based on laws or other legal documents. There is a wide variety of texts that significantly differ from normal texts encountered daily. Content, structure and language are vastly different and more difficult to comprehend. This is not only a problem for every layman who tries to understand those laws but also for a software system processing these texts. There is no universal structure inherent to all texts, which leads to a performance decrease when processing structurally different documents without a ground truth.

There are attempts to simplify tasks for lawyers, judges, practitioners and laymen. The domain of Legal Tech tries to utilize techniques from computer science, statistics and machine learning to enrich data found in the legal domain, e.g. classify documents or legal norms [Wa17], identify entities in a document [GWM18], create additional annotations or references, etc. Introducing systems that are easy to use could help in their everyday work. The automatic detection of textual structures such as sentence boundaries influences the achievable results in different legal research areas and such systems. The structural understanding, as part of the domain knowledge, significantly effects the performance level.

Researchers reported difficulties with standard **Natural Language Processing** (NLP) tools in the legal domain. [WP10] achieved promising results when extracting rules from regulatory texts but also encountered problems with the Stanford Parser while processing different phrase structures. "Linguistic indicators, structure, and semantic interpretations which interact with domain knowledge" [WP11] are utilized by legal experts, but a computer is not entirely capable of modelling or utilizing those pieces of information. The legal domain poses issues for automation of legal work because a "large amount of

linguistic, domain, and communicative knowledge" [Mo04] is needed for tasks such as information retrieval.

In those applications a NLP pipeline is defined with multiple processing steps, each using the results of the previous one. By using smaller and most of the time simple processing steps, a complex task such as machine translation, information extraction or retrieval can be achieved. **Sentence Boundary Detection** (SBD) is one of the most critical steps when aiming for such a task, e.g. translation cannot be done without the exact sentences.

Researchers and practitioners often expect some distinct form of input to their NLP system. Processing unclean or non-perfect input data, the performance of those systems degrades. For example, **Part-of-speech** (POS) tagging is a more difficult task if no clear sentence boundaries are used as an input to the system. To allow the legal domain to better utilize NLP techniques and automate certain units of labour, groundwork like SBD needs to be done.

This Bachelor's Thesis tries to bring some universal structure to legal texts by introducing a system to segment them into sentences. The legal domain introduces challenges to NLP with its more diverse and complex structures. Researchers and practitioners need to be aware of a possible performance decline when using standard NLP tools in the legal domain. The aim of this thesis is to introduce a SBD system tailored for the legal domain, which can then be used to segment a document into its sentences.

1.1. Motivation

As already stated, the structure and contents of a legal text differ from a normal text and even within the legal domain there is a vast variation of documents. The basic assumptions previously made when dealing with SBD are not always applicable in the legal domain. For example, it is often defined as an abbreviation disambiguation task, which is only a partial solution in the legal domain. Here a sentence is not always linked to dots and other characters denoting an abbreviation.

Legal documents incorporate more diverse and complex structures, which are harder to understand for NLP systems. Lists are typical for legal texts and

degrade the performance of NLP tasks in the legal domain such as norm classification [MW10], thus it is critical to correctly identify them. [MW09] investigated the possibility of automatically modelling sources of law, but as a precondition they addressed the need for structured documents. Otherwise this information needs to be automatically extracted or manually annotated. Furthermore, they investigated the disparity between their modelling performance on lists and sentences. Lists decreased the performance of their system significantly. In [Ma12], sentences are one of the essential parts needed for legal text understanding and sub-lists or large blocks of quoted texts introduce issues. Poor SBD performance on legal documents is also reported for LUIMA, a system for legal text processing, but not further investigated [Gr15]. This might be one reason for their relatively low performance levels on creating annotations for legal text at sub-sentences levels.

Recent research has shown that a performance problem exists when dealing with sentence boundary detection even with already existing datasets. In their paper, [Re12] showed the difference between the expected performance and the actual performance on those texts, which is sometimes far from perfect.

With the amount of legal documents produced each day, the idea to automate certain tasks such as creation or retrieval of relevant information quickly arises. For example, the relevant pieces of information for a publisher are the individual parts, paragraphs, sentences, list entries and so on found in the legal documents. In order to enrich the document with a commentary, it first has to be segmented into those parts, which is the direct application of a sentence segmentation algorithm. So far, there is no tool for German legal documents available, thus this task is done in a time-consuming manual process by the publishers of online databases.

The hypothesis is that sentence boundary detection is no perfectly solved task and with rising complexity there might be a need to re-evaluate the previous sentence segmentation system.

1.2. Aim of the Thesis

1.2.1. Research Questions

The following research questions guide the work done for this thesis.

- What are sentences in the legal domain?

This question is critical in order to create the annotations and implement the methods. Sentences are the basis for many machine learning tools or methods and thus also shape the direction this research is moving to. A practical and a theorical balance needs to be found. A precise definition is needed to avoid any edge cases. An attempt to answer the question or more accurate a definition of the rule set to decide which textual units are considered as sentences can be found in chapter 2.

The rules need to be precisely defined before creating the dataset to avoid further needed effort for reworking the annotation. The evaluation of different systems can only start after the annotation of the sentences.

- How should the document corpus be build?

First, the documents need to be chosen. There is a huge amount of legal texts available from laws, judgments, contracts, etc. Regarding the scope of the thesis, the focus will be on a smaller number of more prominent document types.

- Which methods are state-of-the-art solutions in other domains?

In order to answer this question, a thorough literature research is conducted. Afterwards, the results are used to determine the methods for testing on documents from the legal domain. The choice needs to be made in a careful manner because different definitions and scopes for sentence segmentation exist and some might not be appropriate for this domain. Each definition concludes in different approaches for solving the sentence segmentation problem. Overall those methods pose an adequate starting point for this thesis and it is important to evaluate their performance on German legal documents.

- What are the best methods for SBD on German legal documents?

Based on the state-of-the-art solutions in other domains, a tailored sentence boundary detection system for the legal domain should be created. Therefore, the current method with the highest performance needs to be determined and used as a reference value.

Possible methods are the ones found in the related research, any sequence labelling model and a combination of both. This is the crucial question the thesis is trying to answer.

- What are the functional/non-functional requirements of the SBD system?

The requirements are needed to allow usability of the SBD system later on. It is important to define how the system operates and how it can be incorporated in a different context than sentence segmentation. The modules created for this thesis can be used for preprocessing of legal texts in a NLP application.

- How good are existing approaches on German legal documents?

The answer to this question is needed to assess how much the performance of a system could improve when using the tailored segmentation methods.

- Are different solutions required for different legal document types?

There is a huge variety of different legal documents with diverse structures and vocabulary. Thus it might not be possible to create one method which can be reliably used for every legal document type. In order to test this hypothesis, more document types shall be collected and the performance of the SBD system shall be evaluated on those documents.

1.2.2. Research Approach

The whole research conducted is done in an iterative approach. After the literature research, the different methods are conceived, implemented and their performance evaluated. In order to better understand the current methods used in the legal domain, a legal expert from the *Dr. Otto Schmidt* publisher is interviewed. Based on this input, the usefulness of such a system for a publisher in the legal domain is evaluated.

The artifacts produced in this thesis are:

- a modular SBD system for German legal documents with state-of-the-art performance
- evaluations of already existing SBD solutions in the German legal domain
- an overview of the different methods for SBD in legal documents and their SBD performance
- an interactive annotation system
- different evaluation routines: graphical- and textual-based

1.3. Structure of the Thesis

This thesis is structured in four major parts. Firstly, this thesis focuses on the problem statement in connection to the German legal domain and the used definition for sentences.

Secondly, the created dataset is described.

Thirdly, the different solution concepts are explained in combination with a literature review. Based on the results, a SBD system is conceptualised and implemented.

Fourthly, the different modules of the system are evaluated and compared against one another. Their strengths and weaknesses shall be discussed.

To conclude this thesis, possible future work as well as the application of this thesis and results obtained will be discussed.

1.4. Prerequisites

All essential prerequisites needed to understand certain parts of the thesis are explained when needed. Nonetheless, the reader is expected to have basic knowledge about German legal documents and their structure. Mathematics and probability theory are needed to fully understand many of the methods discussed. Knowledge about Neural Networks and basic machine learning would

be helpful, but is not needed to understand the overall findings of this thesis. To understand the rule-based approach a basic grasp of regular expressions is needed.

2. Problem Definition

Sentence Boundary Detection (SBD) or sentence segmentation is the detection and possibly segmentation of text and documents into its individual sentences. Particularly older research papers linked sentences to the existence of a dot [RR97] or used other narrow definitions [Mi00]. A sentence boundary can be found anywhere in the text and is not limited to a textual position surrounding sentence-terminating characters, such as dots. In order define sentences in a legal context, a general definition of sentences is needed first. Then the structure of a legal document is investigated to assess its applicability.

In the following paragraphs, we will often use a concept called linguistic or textual unit. They resemble the collection of a number of tokens or words of arbitrary size and can denote any textual data from a couple of words to multiple paragraphs. Thus this is a recursive definition where each textual unit can be further split up into smaller units.

2.1. Definition Sentences

A **sentence** is often defined as a linguistic unit of multiple words linked in some grammatical way or "a sequence of tokens [...] that together form a full stand-alone grammatical sentence" [SA17b]. Translating this to the legal domain is no trivial task. There are a lot of different textual structures which do not follow this definition and need to be individually assessed and then decided upon, whether they resemble a sentence or not.

Viewing punctuation as "a set of non-alphanumeric characters that [...] provide information about structural relations among elements of text" [Nu90], sentences are seen as some form of encapsulated structural information. Punctuation is used to split language into digestible pieces, thus it should be directly included in the sentence boundary decision.

In most cases, the decision boundary is whether a structure is a stand-alone linguistical unit or not. Or there is a need for some segmentation for a more precise understanding of the text. A set of rules was carefully introduced to annotate every structure found in the legal domain accordingly. This rule set is closely related to the rule set defined by [SA17a] but is now tailored to German legal documents. Another variation of such rules can be found in [De12]. For a well build rule set, the structures found in legal documents need to be analysed in the following paragraphs.

2.2. Analysis of Sentences in Legal Texts

Most of the time, legal texts are structured in smaller parts such as paragraphs, clauses, etc. Their content is often only loosely linked to the other parts of the text. Sentences are long and often incorporate complex structures. The following paragraph from the German Civil Code **Bürgerliches Gesetzbuch** (BGB) stresses one of the problems when dealing with sentences in legal documents.

> §253 Immaterieller Schaden
> (1) Wegen eines Schadens, der nicht Vermögensschaden ist, kann Entschädigung in Geld nur in den durch das Gesetz bestimmten Fällen gefordert werden.
> (2) Ist wegen einer Verletzung des Körpers, der Gesundheit, der Freiheit oder der sexuellen Selbstbestimmung Schadensersatz zu leisten, kann auch wegen des Schadens, der nicht Vermögensschaden ist, eine billige Entschädigung in Geld gefordert werden.

In this case, the grammatical sentences are the clauses (1) and (2), but not the headline. The headline needs to be segmented as well, because such structural information cannot be combined with the other sentence before the paragraph or the first clause. Otherwise a segmentation system would produce many impure sentences with the mixture of such distinct textual units. Distinguishing structural information from normal text is a significant challenge in the legal domain.

In order to understand the basic building blocks of a legal document, the BGB was analysed to collect the different textual units found in German legal documents. Those units were later on refined when encountering new structures in other texts. To summarize the results, the BGB consists of approximately 11,280 sentences, structured in around 2,400 paragraphs. Appendix A.1 shows a more detailed list of the more complex paragraph types which are further analysed in section 2.3 to create the rule set for sentences. All the paragraphs not listed in the appendix are similar to §253, as they are also structured with multiple clauses often consisting of a number of sentences per clause. The different structural classes are mostly based on the differences between paragraphs such as list structures and enumerations.

In this thesis, a textual part of a legal document is seen as a **sentence** if its combination with the next or previous sentence destroys that sentence's meaning or adds unrelated pieces of information. This is the basis of all the structural decisions made in the following section, though the definition makes SBD more difficult. This way sentence boundaries can potentially be found at any location in the text.

2.3. Sentence Boundary Rule Set

When deciding on the different rules for a sentence, the most important thought taken into consideration is to keep sentences intact and not add additional unwanted tokens. Those textual pieces would destroy the grammatical order or meaning of a sentence and thus need to be segmented on their own. The following categories are heavily influenced by [SA17a] but adapted to the German legal domain. To define a sentence boundary rule set, the different types of sentences need to be analysed and listed:

Most of the sentences found in legal documents are **normal sentences**, i.e. they have a subject, object and verb in the right grammatical order. Additionally, they are concluded by a sentence-terminating character such as a period, semi-colon, question or an exclamation mark. In this case, everything between two sentence-terminating characters is seen as a sentence.

Another structure very closely related to normal sentences are **linguistically transformed sentences**, which are in some way altered, but still have the

basic building blocks of a normal sentence. By changing the word order or making other small adjustments, these sentences can be transformed into a normal sentence. They are treated in the same way.

There are more complex structures in legal texts all collected in one category, the so called **special sentences**. Those are linguistic units with stand-alone content that cannot be incorporated into other sentences. If combined with other sentences, they would destroy the completeness of those sentences, i.e. grammatically correct sentences would be altered to a semantically wrong form. The following paragraphs describe those different textual units and whether they are considered sentence boundaries in legal documents. All following structures are an individual part or segment of the text and are annotated as a sentence if not stated otherwise.

Headlines are one of the most common textual units found in legal documents. They are used to determine the structure of the text and to show which parts of the documents are related to another, thus they convey crucial information to understand the overall structure of the text. Sometimes, they are also referred to as free-standing lines, a line ending without punctuation but still concluding a semantic concept.

Citations are often part of judgments and thus need to be addressed in this thesis. They are used to link different documents and passages. Citations in the middle of a sentence are not annotated (if they are not a sentence on their own), otherwise they are normal sentences with additional tokens at the beginning and the end. **Ellipses**/**Parentheses** are treated in the same way.

One of the most obvious structures in legal documents are **lists**, which have a wide variety of different types each with its own consideration. First, lists of **alternative sentence ends** or stand-alone sentences convey no message beyond their own list entry. Thus, they are annotated as individual sentences. Second, **lists of lists** are treated in a recursive way and the decision on the sentence boundary is based on the sub-entries. If the entries consist of multiple sentences, these are also annotated. As [Ma12] states, lists can form "single sentences [...] [though] the list items are often referenced separately". This more fine grained segmentation of those legal structures simplifies further processing and is motivated despite the fact that thus incomplete sentences can be created.

Another variation of lists are **enumerations** consisting of short entries, which lack a verb or a concluded operation. In sentences with a variable middle part, i.e. a list is used to denote the different subjects or objects of a clause, the list items are not individual sentences but form one whole sentence because they do not conclude any action. All of those list variations may start with a sentence concluded by a colon. Otherwise the introductory textual unit is not annotated as an individual sentence.

Definitions are a combination of a headline and (multiple) sentences, thus their individual parts are annotated accordingly. **Data fields** which hold the information about dates, people, locations, etc. and case names are similar. It is illogical to combine the different pieces of information, thus they are segmented separately. **Endnotes**/**Footnotes** are also considered as a combination of headline and sentences.

Page numbers are, if not already removed from the text, only annotated if they are found between two sentences.

When reading through the text, **mixed cases** of all the previous structures can be found. The described textual units are only the basic building blocks found in a legal document. There is a large amount of variation even within the same legal document type. In such cases, the smaller structures are annotated, e.g. in a combination of enumerations and lists, the enumerations would also be annotated as sentences to avoid any inconsistency. Those structures need to be considered when further processing. Possibly, they need to be restructured based on the use case.

Sometimes, typographical errors were encountered and removed as good as possible. In order to allow better processing later on, structural information linked to a sentences is kept with the sentence, e.g. the clause number in a paragraph is always part of the first sentences of that clause. Thus it is possible to also reassemble a piece of text in the way wanted.

Regardless of type, smaller sentences are preferred over larger structures. Firstly, this is in relation with the accumulation of errors. A translation system can achieve good performance even if the sentences are split to much. In comparison, if multiple sentences are combined into one sentence, the translation is insufficient. Secondly, smaller parts allow a computer system to reassemble

a given text structure. It is easier to decrease the granularity of the segmentation. For example, manipulating list structures is straightforward when they are segmented into individual parts. Thus smaller structures are preferred in general and can, if not wanted, still be combined to produce more complex structure such as paragraphs.

The aim was not to create the perfect definition for a sentence in the legal domain but to create a logical, consistent and practical rule set to work with. This thesis sometimes produces textual parts which cannot be described as complete sentences in a grammatical way because they are lacking essential parts of speech. Those parts are needed for further processing steps, as overall they reduce the complexity of the given text part.

3. Legal Dataset

The two main datasets used in almost all research concerning the topic are the Wall Street Journal (WSJ) corpus and the Brown Corpus. WSJ consists of English newspaper articles which are segmented into their individual sentences. The Brown corpus is a more wide spread dataset with around 500 documents from 15 different text genres [Gi09]. Additional datasets sometimes used are the CDC dataset, a collection of various Sherlock Holmes stories with 5,692 sentences, and the GENIA Corpus, 16,392 sentences from biomedical research documents [Re12]. As seen later in the related work section, nearly perfect results can be expected on those datasets because they are well researched and mostly understood. A dataset needs to be created for this thesis to assess the performance of SBD systems in the legal domain. Two challenges arise from conducting research on the traditional datasets, which will be shown in the following paragraphs.

Firstly, there is a language barrier. English is the most common language for SBD datasets. English and German have different sentence structures and usages of punctuation. The second problem is a structural one. Legal documents and the commonly used newspaper articles are vastly different document types and even within the legal domain, there is a much larger diversity when comparing documents such as laws and judgments. Therefore a custom dataset for the legal domain is required.

The corpus consists of a diverse set of different German legal documents. The main focus is on judgments and laws with around 20'000 sentences each, but privacy policies and terms of service are also collected. In the dataset there are approximately 55'000 sentences. The judgments and laws are used for training the different methods and the other document types mainly for validation and testing.

The aim of this dataset is to have a wide variety of documents in order to better approximate the performance of a SBD system operating in the legal

Document type	# Documents	# Sentences	# Tokens
Laws	13	20,322	498,403
Judgments	131	21,484	518,052
Terms of Service	100	8,297	175,529
Privacy Policies	11	2,186	53,552
Wikipedia	14	2,129	43,638
XML	2	1,627	80,453
Total	271	56,045	1,369,627

Table 3.1.: Statistics on the dataset created per document type.

domain. BGB, **Grundgesetz** (GG), **Sozialgesetzbuch** (SGB) 1 to 3 and the **Strafgesetzbuch** (StGB) are included in this dataset, with the BGB being by far the longest text. The judgments are collected from the website Bayern.Recht[1], a collection of many judgments from Bavaria, and are from different legal domains and courts (Verfassungs-, Ordentliche, Verwaltungs-, Finanz-, Arbeits- and Sozialgerichtsbarkeiten). The privacy policies of major tech companies are collected and the terms of service of online shops gathered by David Koller in his Master's Thesis [Ko19] are also used. A detailed statistic on the documents and the number of sentence boundaries can be seen in table 3.1 and all the used documents are listed in appendix A.1.

The dataset has a comparable size or is even bigger than most standard SBD datasets. This size is needed for a reliable model training on each document type and to assess a possible difference when processing different legal documents. The main parts of this dataset are laws and judgments, which have reasonable size to allow stable model fitting and significant training results. The other parts of the data collection have approximately the size of the test or validation sets used during training with around 2,000 sentences.

Additionally, a small corpus of German Wikipedia articles and some laws in a XML file-format are collected to assess the performance of the SBD system on those test cases. This was done in order to see how a change of corpus affects the different systems and asses their robustness. Additionally, the usefulness of annotation in a legal document will be assessed.

The documents are annotated with the help of an already existing online annotation tool available at the chair and later on with a newly created **Graphical**

[1]See https://www.gesetze-bayern.de/

User Interface (GUI) which is also used to display the classification results. The last non-special token or word in every sentence is annotated as the end of the sentence. This decision is a practical one considering the possibility of removing the special tokens or additional cleaning without the need to reset the annotations. It would be more difficult to do that if the annotations were placed on the sentence-terminating character such as a dot which is often removed in a NLP pipeline. Another reason is that the system would have to distinguish between sentences that end on sentence boundary characters and sentences that end on words which could add a layer of confusion to the task.

4. Solution Concepts

Before introducing the created system for SBD tailored to the legal domain, the different methods previously used for sentence segmentation need to be discussed. Generally, the task of sentence boundary detection or sentence segmentation is solved with three different approaches: rule-based methods, supervised and unsupervised machine learning. These are all linked to the respective definition of a sentence. Even in previous research there are narrower and wider rule sets for sentences, which obviously lead to different solution concepts.

Rule-based approaches use hard-coded rules to detect sentences. Those are often tailored to a very specific type of document or domain and are the most labour-intensive methods but achieve exceptional performances in their domain. Regular expressions, lists of abbreviations, common sentence boundary words as well as lists of names and institutions are often used in this case. One example is the Alembic system which utilizes various preprocessing steps and rules [Ab95]. Further approaches exist to automatically extract rules for sentence boundary detection as discussed in [SFK99].

Supervised machine learning approaches use different types of statistical models for the relation between words and sentence boundaries. **Maximum Entropy Models** (ME) [RR97], **Hidden Markov Models** (HMM) [JW13] and **Neural Networks** (NN) [PH97] were previously used for this task. Newer techniques are **Conditional Random Fields** (CRF) [Su18][LMP01] and Recurrent Neural Networks, which are generally experimented with as sequence labelling models [Sh18].

Unsupervised approaches try to solve the task of sentence segmentation solemnly based on the text documents without annotations. This means they want to distinguish for example sentence boundaries from abbreviations, which are also concluded by a dot. In this approach hypothesis testing and frequency analysis are used, see e.g. [KS02a].

To get an overview, [Re12] researched sentence boundary detection as a whole. One aspect investigated is the dataset used for performance evaluation. They argue that the problem is oversimplified if it is only defined as a binary classification on sentence boundary characters, and the testing environments are vaguely specified. For them, sentence boundaries are not directly link to punctuation marks. They tested the performance of different available tools on the corpora Brown, CDC, GENIA, WSJ and the combination of all corpora and found "low performance levels for some of the tools" and a lacking "robust[ness] to corpus variation" [Re12]. The best performance without additional preprocessing was an F1 score of 96.8% on the Brown corpus, 98.6% on CDC, 99.6% on GENIA and 99.2% on WSJ. The performance on their test corpus for User-Generated Content was significantly worse. They express their concerns about the expected performance, which is sometimes far away from previously assumed performance levels above 99%. They propose the usage of mark-up processing to increase the performance on noisier user-generated content.

4.1. Related Work on SBD

In the following paragraphs, a temporal overview of the SBD research conducted will be given. This way, the evolution of sentence boundaries and methods is expressed. Some papers are hard to compare performance-wise because of a steady enhancement of definition.

[PH97] used the **Part-of-Speech** (POS) for every token as the input to different machine learning classifiers such as decision trees and neural networks. They created the required input information via a small POS-lexicon[2]. As a heuristic for unknown words, the so-called Satz system assigns equal probability for all possible POS classes. Based on the POS-distribution of the token to classify and its context, the system achieved an error rate of 1.1% on the WSJ, 1.3% on the newspaper Süddeutsche Zeitung and 0.7% on the German News Corpus[3] with the NN. With the decision tree, an error rate of 1.0% on WSJ, 1.9% on Süddeutsche Zeitung and 0.7% on German News was achieved.

[2] They approximated a size of around 5,000 words for the POS-lexicon and 300-500 sentences to achieve a performance surpassing other exisiting solutions [PH97].

[3] Those datasets are build similarly to the English newspaper corpus WSJ, but only use German newspaper articles.

[RR97] used ME models with different features as input and lists such as abbreviations constructed from the training data. For them, sentence boundary candidates are words directly followed by a dot, exclamation or question mark. Thus, they try to distinguish abbreviations from normal sentence ends, a task called period disambiguation. Predictions are based on the features of the candidate and the neighbouring words. They reported an accuracy of 98.0% on the WSJ Corpus and 97.5% on the Brown Corpus.

Another unsupervised approach focusing on period disambiguation was introduced by [Sc00]. Their method calculates the frequency of the token appearing within a certain context, such as number of times the token is followed by a dot and a captialized word. Based on those frequencies, the probabilities for the different context types are estimated and the one with the highest probability is assigned as the prediction. In their evaluation, they achieved an accuracy of 99.6% on WSJ and 99.8% on Brown.

[Mi00] investigated the possibility of incorporating the sentence disambiguation into the POS tagging phase, thus also use POS information to separate the sentences. HMM and ME are trained on a tri-gram POS tagger framework, using the token with its two adjacent tokens for a classification. In their work, a sentence boundary can be found "on periods and other sentence-ending punctuation and on word-tokens in mandatory positions" [Mi00]. Mandatory position denote the need for a capitalized word after the usage of punctuation. In their experiments, embedded sentences such as citations were treated the same way as normal sentences. The system can also be trained in an unsupervised way. Their best performance on the Brown Corpus was an error rate on sentence punctuation of 0.20% and 1.87% on words in madatory positions. The respective error rates on the WSJ Corpus are 0.31% and 3.22%.

In their following paper, [Mi02] introduced further heuristics and rules used to identify abbreviations. They used common words, common words for mandatory positions, proper names, a list of abbreviations and a list of common sentence starters. Overall, their definition for sentences revolves around the disambiguation of capitalized words. To do so, they used a rule set in combination with their heuristics, lists and a decision tree achieving an accuracy for the disambiguation of sentence breaking punctuation on the Brown Corpus of 99.7% and on the MUC'7 Corpus of 99.3%. In their opinion, the identification of abbreviations "largely [solves] the sentence boundary problem" [Mi02].

[KS02a] tried to detect abbreviations with a scaled log likelihood ratio which is based on the number of occurrences a token is found before a dot versus not. In [KS02b], they define SBD as collocation identification, thus focusing primarily on the detection of abbreviated words. In 2006, [KS06] expanded their model with different heuristics such as a list of "frequent sentence starters" which resulted in the creation of their Punkt system. In contrary to other researchers, they also concentrated on the German language. The Punkt system is part of the NLTK framework tested in this thesis. On many of their multilingual datasets, they reported a performance above 99% for almost every metric used.

A SVM model with a linear kernel to distinguish between sentence ends, abbreviations and abbreviations which are also sentence-terminating was used by [Gi09]. Their research mainly focuses on the disambiguation between sentence boundaries and abbreviations, with a small number of examples in the overlapping region. The prediction is based on the context around a dot and the features of those words. Their error rates were 0.25% (WSJ), 0.36% (Brown) and 0.52% on the poetry corpus (Poe).

[JW13] investigated the performance of HMM for sentence boundary disambiguation based on different features such as the length or case of a token. They use a more flexible definition for sentence boundaries and even incorporated headlines. Combining a rule-based scanner with their HMM, they reported an F1 score of 96.0% on the German corpora TIGER and 98.5% on the WSJ corpus.

The performance of CRF on German text documents was evaluated by [Su18]. They use a very rich feature set, using POS information, class information based on the clustering of word embeddings etc. Processing a postcard corpus for evaluation, they achieved a F1 score of 95%, whereas Punkt achieved an F1 score of 79%.

4.1.1. SBD in the Legal Domain

There is a lot of research conducted in the general field of sentence segmentation, in contrary to the legal domain and especially with German legal documents. [SA17b] investigated the performance of Conditional Random fields on

American court decisions in contrast to the performance of the already existing systems CoreNLP, OpenNLP and Punkt tested. They reported the existence of a performance gap between classical sentence boundary detection and sentence segmentation in legal documents. In [SA17a], they further described the CRF used and their annotation scheme for sentences which influenced the one created in this thesis. The highest F1 score achieved on their legal dataset with an existing solution and additional preprocessing was 89% by OpenNLP, although the average performance is lower. In contrast, their average F1 value is approximately 95% when using their custom trained CRF. The scores are significantly lower than reported by the researchers in other domains like newspaper articles.

4.1.2. SBD in the German Legal Domain

The issue with all this research is twofold. Firstly, there is a lack of comparability. Documents from the legal domain follow a different structure and sentences may not even end with a sentence-terminating character. Focusing on well-structured datasets and problem definitions, a nearly perfect solution for those problem statements could be achieved. When investigating further, there is a gap between the performance expected and delivered by the systems and concepts in use. Only in previous years, researchers seem to focus on other datasets with different challenges seen by the performance decline of standard SBD systems on their datasets. The performance results above 99% are far from realistic and not based on more complex, real word examples.

Secondly, previous research on sentence segmentation has a very narrow definition for sentence boundaries, for example only revolving around period disambiguation, which is not applicable here. When segmenting a legal text only based on abbreviation detection, the system will produce impure, mixed sentences by combining headlines, citations and lists with normal sentences.

While conducting this thesis no research about SBD in German legal documents was found and most papers investigating SBD in German texts focus more on a wider, multilingual applicability of their approaches. No statement about possible performance levels of the different methods can be made, although CRF have the highest performance levels and outperform existing solutions. In recent years many state-of-the-art results in various disciplines

were achieved by NNs, thus their application possibilities will also be investigated. The question arises, which performance levels can be achieved for SBD on German legal documents? Before attempting to answer this question, the system and methods used in this thesis need to be introduced.

4.2. Sentence Boundary System

In order to segment legal texts later on and also test different approaches, a SBD system is implemented. It consists of multiple parts, each described in the following sections. An overview can be seen in 4.2.5. The functional and non-functional requirements for the SBD system are described below.

A target for the created system is to have a high level of modularity in order to achieve an flexible testing environment. There are six different modules for sentence segmentation which either use the whole text or the already tokenized text. The output of each module is a set of predictions for the sentence boundaries and every input token. Those can then be used to segment the sentences or the tokens into the individual parts.

4.2.1. Functional/Non-functional Requirements

The system should:

1. Be usable in combination with other NLP systems and platforms which often use sentences or even a tokenized text as an input
2. Accept text documents or tokenised text as an input and output sentences as a list of strings or a list of the individual tokens [4]
3. Allow to create additional annotations and edit already annotated documents via a graphical user interface
4. Allow to choose different modules for SBD
5. Produce predictions for sentence boundaries based on the chosen prediction module

[4] The output format is similar to the ones found in existing systems. This allows for an easier integration with existing frameworks.

6. Give visual or textual feedback as an evaluation for the chosen prediction module

4.2.2. Non-functional Requirements

The non-functional requirements for the implemented system are:

1. Program code should be written in Python.
2. Different modules shall be implemented with state-of-the-art frameworks and libraries.
3. Abstraction principles shall be used when implementing the different modules.

4.2.3. Tokenisation

Based on those requirements the different parts of the system will be build. First, the input, given as a string or read from the file location, is separated into its distinct text parts, called tokens. Small textual units, such as words or special characters, are often used when processing text in NLP. That means the text is split up into its smallest parts to gradually rebuild it in the way required.

For the system an aggressive tokenisation strategy is used, also utilized in other research [SA17b]. All escape sequences except the newline sequence is removed because it might indicate the end of a free-standing line. Words, numbers and special characters are separated from one another. For example, abbreviations are not processed in combination with their terminating dot, thus making no previous, maybe misleading assumptions. This allows an easier disambiguation between abbreviations and words which terminate a sentence. Additionally, the diverse structures found in legal documents can be easier and in a more general way processed. There are many ways legal documents are written and in most cases it is not possible to determine every possible writing style and combination of special characters.

The tokenisation is one of the preprocessing steps of the system. On the one hand, smaller tokens lead to larger machine learning structures, because longer

token sections are needed to process the same text. On the other hand, smaller tokens might allow the system to generalise and model the overall structure in every document better. Closing brackets always finishing a semantic concept, no matter the context.

4.2.4. Graphical User Interface

In order to understand the different prediction modules better and improve the testing and user experience, a **Graphical User Interface** (GUI) was implemented to show the produced predictions. This interface is also used to produce additional annotations. Documents can be created and loaded from disc.

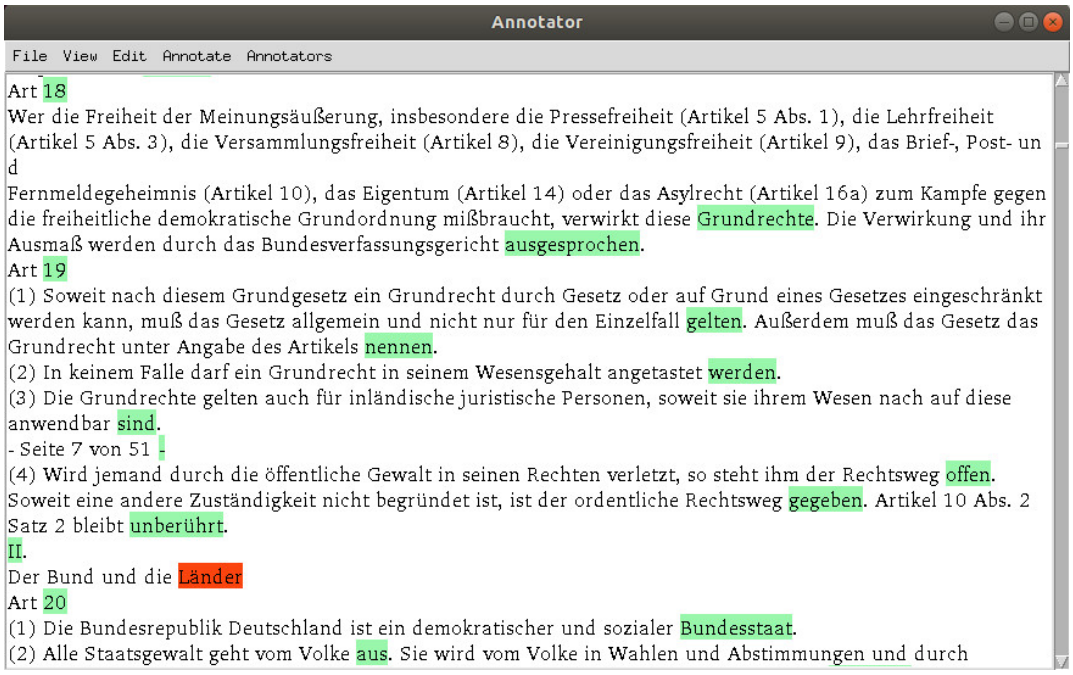


Figure 4.1.: GUI for creating annotations and testing the modules in a graphical fashion

It is possible to chose different annotators and then calculate the predictions for the loaded text. All annotated sentence ends are highlighted in red when loading a text. The true predictions are coloured green, wrong predictions are black. Sentence ends not found by the prediction module are still coloured red. The color coding can also be seen in figure 4.1.

This way, it is easier to see the strengths and weaknesses of the different modules, their working performance and observe the influence of changes to them. The GUI is particularly useful for the creation of the CRF and rule module. Every new rule or feature manifests themselves in some visual changes to the predictions and lacking features can be extracted based on the observed errors.

4.2.5. System Overview

The system itself consists of three essential parts. The first part is a **Tokenizer** module which takes the path to a text file or the text itself as an input. The text is segmented into individual tokens which resemble the words, special characters, numbers, etc. as seen in the tokenisation part. Those tokens are (if tokens are needed) used as an input for the various different prediction modules. The already existing frameworks OpenNLP and NLTK are tested using text as an input, the newly created and trained methods are more variable and accept tokens as well.

Secondly, the prediction modules produce a truth value for each token, whether this token is in fact a sentence boundary token or not. The various ways to produce those predictions are described in detail in section 4.3.

Thirdly, the **Splitter** module is then used to create the output of the whole system, taking the predictions and the text or the tokens as an input. The module then produces the multiple lists of tokens or one string for every sentence. The resulting sentences can then be used for further processing.

This way, variability is achieved, because every individual step has its own parameters and can be swapped with (almost) any other module with only little modifications. By that, not only a highly variable testing environment was build but also a system modifiable to the own use cases.

Figure 4.2 gives an abstract overview of the possible data flow in the system. The Template module can potentially be used as the input to all other modules but cannot produce predictions on its own. The output of the whole system is then returned to the caller, e.g. the GUI or any other program. The seen modules will be described in detail in the following sections.

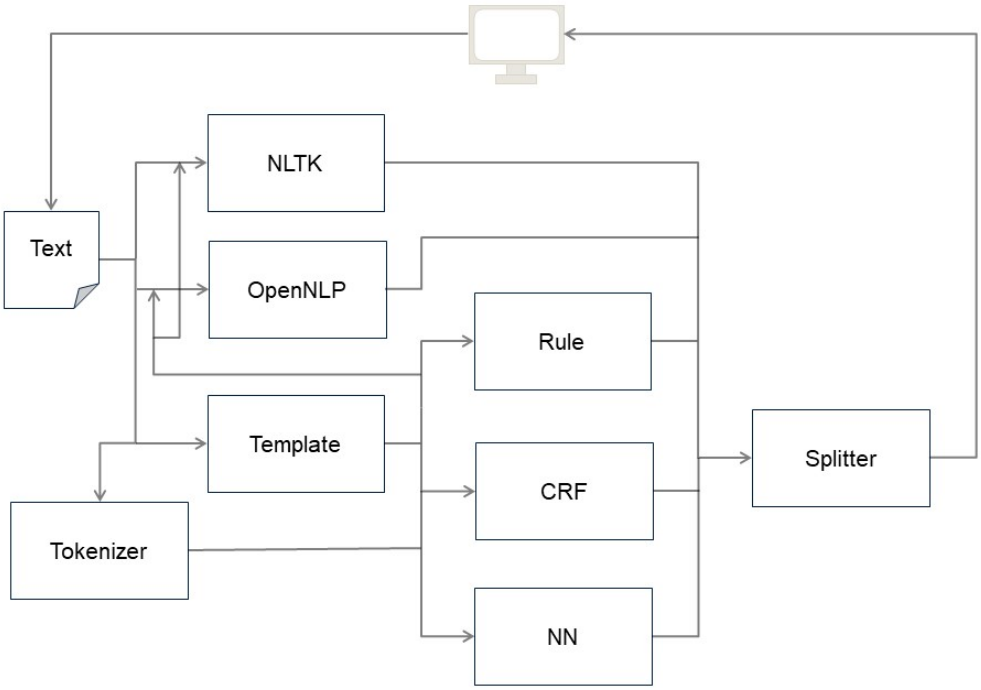


Figure 4.2.: Simplified dataflow through the segmentation system

4.3. Methods

In the following sections, we will discuss the different methods and modules implemented and tested in this thesis.

4.3.1. Rule-based

In the rule-based approach, various rules are defined for sentence boundaries. These are then matched to parts of a given text. A sliding window with different sizes is used to match rules based on the window size to the structure of the text. The window creates a limited view of the text, taking a small number of tokens or words into consideration, e.g. a rule with a window size of three is matched against each sequence of three consecutive tokens in the text. Only if each individual token matches to the corresponding regular expression at their position in the window, a predefined rule is applied to that specific textual position.

There are two types of of rules used in this thesis, namely positive and negative rules and they are written in the following way: The boundaries of the window are given with square brackets. Each regular expression and thus token in the window is separated with commas and multiple possibilities for one token are denoted by a line. More complex regular expressions are abbreviated by their name in italics and can be found in table 4.3. If multiple consecutive tokens need to match the same regular expression, this is also abbreviated by the number of consecutive occurrences after the expression in square brackets.

In the first half of the algorithm, the text is checked against the positive rules, i.e. their corresponding window is placed on every possible position. If each token matches their regular expression, a sentence boundary is placed in the middle of the window. If the window size is even, there is no middle position, thus the boundary is assigned to the following token. The same logic is applied to the negative rules which are afterwards checked against all possible positions and sentences boundaries are removed if there is a match. As an example, we introduce one positive and negative rule:

The paragraph rule [§, *Number*, \n] is used to detect paragraphs from normal text. A window of three tokens is used to match the beginning of a paragraph structure and detect the headline of those paragraphs. Only if the first token in the window is a paragraph sign, the next one a number and the window is concluded by a newline, the rule is applied. The remaining positive rules can be found in table 4.1.

The abbreviation rule [*Abbreviation*] uses a window of one word and checks whether this word is matched to a regular expression containing some basic abbreviations. If the word is found in this list, the possible sentence boundary prediction at this position is removed. The other negative rules can be seen in table 4.2.

The other rules can be found in the following table. This module does not model each possible sentence but is not further refined, because within the scope of this thesis there are more promising solutions. Nonetheless this method has its advantages as errors can directly be addressed by adding new rules. With the other methods there is no direct option to influence specific predictions.

Type	Rule
Normal Sentence	[*AlphaNum*, *AlphaNum*, *AlphaNum*, *Boundary*, *Upper*]
End Citation	[*AlphaNum*, *AlphaNum*, *AlphaNum*, *Boundary*, "] [*AlphaNum*, *AlphaNum*, *AlphaNum*, ", *Boundary*]
End of line	[*, *, *AlphaNum*, *Boundary*, \n]
Paragraph	[§|*Upper*, *Number*, *AlphaNum*, \n] [§, *Number*, \n]
Headline	[\n, *AlphaNum*, \n] [*Upper*, *Upper*, \n] [*, *, *Upper*, \n, *Number*|§]
Dot	[*[3], *AlphaNum*, *Boundary*, \n, §|Art|*Number*] [Comma, *AlphaNum*, .]
Add. Sentences	[*AlphaNum*, *AlphaNum*, *Boundary*] [\n, *AlphaNum*, *Boundary*] [*, \n, *AlphaNum*, *Boundary*, \n]
Abbreviation	[*Abbreviation*, ., *AlphaNum*, *Boundary*]
Parentheses	[*, *Parentheses*, *Boundary*]
Page Numbers	[*AlphaNum*, *Number*, -, \n, *SpecialChar*|§|Art]

Table 4.1.: All positive rules applied

Type	Rule
Abbreviation	[*AlphaNum*, *AlphaNum*, *Abbreviation*, ., *AlphaNum*] [*Abbreviation*]
Number	[*, *Number*, .]
No Headline	[*, *, *AlphaNum*, \n, *Lower*]
Boundary	[*Boundary*]
Enumeration	[\n, *Number*, .] [\n, *Special*, *Number*, *Special*, *Upper*]
Character	[*|\n, *Character*, .]

Table 4.2.: All negative rules applied

4.3.2. Template Module

As legal texts are almost always annotated with numbers for paragraphs and other sections, the template module will be used in addition to the other modules and tries to utlize those structures. It is used to increase the structural performance of the modules, i.e. their ability to correctly identify headlines, paragraphs and lists. This identification is done via a list of regular expressions which match to the beginning of a headline or list. Based on a text, they can be

Abbreviation	Regular Expression
AlphaNum	[0-9A-Za-zÄäÜüÖöß]+
Upper	[A-ZÄÖÜ][0-9A-Za-zÄäÜüÖöß]*
Lower	[a-zäüö]+
Number	[0-9]+
Character	^ [A-Za-zÄäÜüÖöß] \Z
Boundary	[.:;?!]
Special	[.:;?!,#*§$%&()[]{}]
Parentheses	[()[]{}]
Abbreviation	Abs|Art|Rn|Urt|Buchst|bzw|usw|BL|ff|Nr|ca|Hr

Table 4.3.: Abbreviations for more complex regular expressions

manually created, but it is challenging to deal with many different documents. Each document has other ways to denote headlines and lists. For example, the structure of the GG is different to the BGB. In the first one, paragraphs are started with "Art. number" whereas in the second one a paragraph sign in combination with a number is used. The structure differs a lot more when processing other document types from the legal domain such as judgments.

In order to reduce the amount of manual labour needed, an automatic system is introduced to produce the trigger rules before a document is processed. This automatic system is based on the following heuristic: Nearly all legal documents are structured with numbered paragraphs and sentences. The algorithm seeks the first line which contains the beginning of such an structure, e.g. the line which starts with "Art. 1", and constructs a regular expression from it. This way, the text can be segmented into the header information and the main body.

The main text body is skimmed for recurrent patterns at the beginning of a segment, not taking the first line of a segment into consideration. The first line contains the structural information and will not be part of any new segment bounds. If a numbered structure is encountered at the beginning of the next line and the previous line does not end with a character symbolizing the beginning of an enumeration or list such as a comma or a double dot, this pattern will be translated to a trigger matching that structure.

The regular expression is created in the following way: First, the line is tokenized to extract the individual words, special characters and escape sequences.

Each number is translated into a generic regular expression for numbers, each special character and word inserted directly. Spaces and other escape sequences are also added to the regular expression. Because the structured components start at the beginning of the line, the regular expression first has to match a newline character. All the headlines and other structure can potentially be matched.

Every token up to the first non-special character will be incorporated into the trigger. This allows the system to capture a huge variety of different numbered segments. The whole processes is done in an iterative approach, meaning the segments are then searched for recurrent numeric patterns.

For every possible pattern the system will process the entire text and split it up based on the matching text parts as boundaries. Thus, the system can segment the text into is coarser structure.

4.3.3. Conditional Random Fields

Conditional Random Fields are a statistical model introduced by [LMP01] and commonly used for sequence modelling, i.e. assign labels for the items in a possibly correlated input sequence. CRF use the information about previous labels and their features to produce the labels for the given input sequence up to the current data point. CRF [SM11] are random fields for the conditional probability $P(y_{1:n}|x_{1:n})$, with $x_{1:n}$ being the input sequence $x_1 \dots x_n$ and $y_{1:n}$ the output sequence $y_1 \dots y_n$. The special case of linear-chain CRF use ME models and normalize the probability globally in a sequence. The output depends only on the current features and the previous label, seen in the graphical depiction in 4.3. The edges depict dependencies and the label inference in a CRF is only numerically tractable when a tree or chain structures are formed by the dependencies. The labels are then inferred via the following probability:

$$P(y_{1:n}|x_{1:n}) = \frac{1}{Z(x_{1:n})} \exp(\prod_{n=1}^{N} \sum_{d=1}^{D} w_d f_d(x_{1:n}, y_n, y_{n-1}))$$

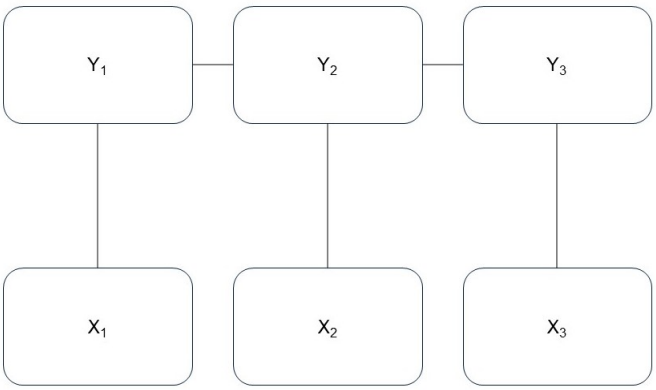


Figure 4.3.: Linear-chain CRF

$$Z(x_{1:n}) = \sum_{y_{1:n}} \exp(\prod_{n=1}^{N} \sum_{d=1}^{D} w_d f_d(x_{1:n}, y_n, y_{n-1}))$$

Linear-chain Conditional Random Fields are comparable to a sequential form of logistic regression, different features are mapped via an exponential function in combination with a coefficient to the prediction domain. In this case, a linear-chain CRF predicts based on the features for every token whether they are the sentence boundary token. They can be seen as a generalization of HMMs.

For the implementation of this module CRFSuite [Ok07] is used, which allows a flexible feature definition and various training schemes. The list of tokens is translated into a list of features for each individual token via the abstract class of feature extractors. The surrounding tokens and their features are also taken into consideration.

The features for a token are a combination of their own properties and the specific properties of the tokens in the predefined window around the word or token. The size of the window is variable for each feature extractor used. A list of all extractors and features used can be seen in the table 4.4.

Feature Extractor	Output
Feature	Adds a bias to the features
Title	Adds a truth feature if the token is the first in the window and uppercase
Length	Extracts the length of every token in the window. Also used to distinguish between possible abbreviations
Signature	The signature of each token is extracted, which is as a feature a combination of the length and the type of the token, e.g. numbers, words and special characters. The token is then translated into the specific combination of numbers as "N", lowercase characters as "c", Uppercase characters as "C" and special characters as "S"
Lowercase	Extracts the individual tokens in lowercase. This way, it is possible for the system to learn certain trigger tokens, which denote the end of a sentence
Combination	Uses to different feature extractors. Only if the first extractor has a positive output, i.e. not "false", the feature of the second extractor will be added[5]
WordsPerLine	Extracts the number of tokens on the line the specific token can be found. There is often a difference between the number of tokens in a headline versus the number of tokens found in the normal text. Thus shorter lines would be an indicator for headlines
Number	Extracts wheter the token is a number. Also useful to determine whether the token can be found on a headline or not

[5]The idea behind this extractor is the possibility to add simple rules for feature extractors such as: Only if the token starts with an uppercase letter, output the length of that token. This rule could potentially be used to detect abbreviations which are most of the time shorter tokens.

Special	Categorizes the different special characters. If a token is not a special character, the output is "No", otherwise "End" stands for sentence-terminating characters, "Open" for opening parentheses, "Close" for closing parentheses and "Newline" for newline characters. In all other cases, such as the paragraph sign, the output is "S". The output of this extractor is important as the combination of different characters and tokens often denote recurrent structures, e.g. headlines, lists or even normal sentence ends
Upper	Extractor which outputs the truth value if the token starts with an uppercase letter to distinguish abbreviations from sentence ends. If the token after the dot is uppercase, this is a strong indicator for a sentence end
Lower	Same idea as the Upper-extractor, but in this case the output is "True" for all tokens that start with a lowercase letter
SBD	Outputs the truth value if the token is part of the sentence-terminating characters
Parentheses	Distinguishes between parentheses and other tokens
Newline	Distinguishes between newline characters and other tokens
SBDNewline	Distinguishes whether the last sentence boundary character is directly followed by a newline
Abbreviation	An manually defined abbreviation list is used in the feature calculation. Distinguishes whether the token is in the predefined abbreviation list

Table 4.4.: The possible features usable as input for the CRF

Many different combinations of features were tried. Based on the average F1 score for a feature, the respective window and features are chosen for further testing. Due to the broader scope of the thesis, only the most promising features were chosen and combined to form a simple model. This iterative procedure showed that the most important features with the highest average F1 scores were the signature of a token, whether the token was a special character

and especially the token in a lowercase form. The conclusion for this is that the CRF learns certain trigger words or signatures which denote the end of a sentence. Special tokens in legal texts often convey structural information such as the beginning of a paragraph or list. A more detailed evaluation of the performance can be found in section 5.3.

4.3.4. Recurrent Neural Networks

NN are one of the most prominent models used in machine learning. They consist of many simple processing units or neurons, which all have their own properties, weights and behaviours. A combination of many neurons is called a neural network, capable of modelling complex input and output behaviour. To train this model for a certain task, additional data is continuously fed and the output of the whole network is compared to the required output behaviour. Afterwards, the weights are updated a tiny bit, gradually pushing the network to the wanted predictions. This is performed by a procedure called Gradient Backpropagation, where the weights of an individual neuron are changed based on its influence on the whole output behaviour. Generally they are structured in layers, with the output of the previous layer used as the input for the next one (see 4.4). Each neuron takes an weighted input of the previous layer and combines it with a non-linear activation function such as the sigmoid. The NN can be optimized to approximate the wanted output. Commonly used are linear or convolutional layers as the basic building blocks of a NN.

In this case, a **Recurrent Neural Network** (RNN) is trained, which has the ability to keep certain pieces of information over a longer amount of time. Those networks consist of multiple recurrent units, which have additionally to their normal input a hidden state. This hidden state is kept and updated for every input token in the steady stream of data. As seen in figure 4.5 the hidden state can be used in the next processing state. Sequential data is often better modelled by recurrent neural networks, because normal NN do not have the capability to save information for a longer period of time.

All neural networks tested are implemented in PyTorch [6] and consist of bidirectional layers of a recurrent neural unit such as RNN, LSTM or GRU, which is followed by linear neural layers or convolutional neural layers in combination

[6] Open source deep learning platform, see https://pytorch.org/

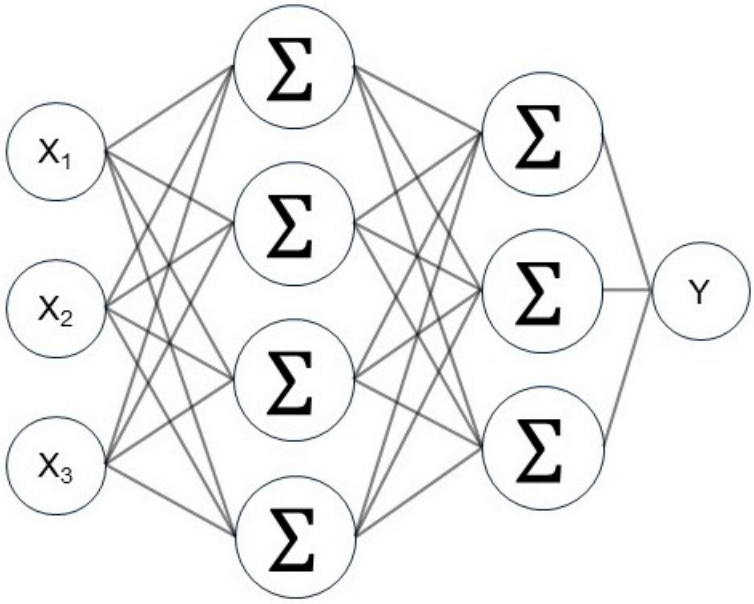


Figure 4.4.: A two-layered Artificial Neural Network

with linear layers. The sigmoid function is used to produce the wanted output whether the token of choice is a sentence boundary or not. The non-linear activation function between the neurons is a rectified linear unit. A symmetric window of tokens is used as an input for the whole neural network with the possible sentence boundary in the middle.

The whole neural network receives word embeddings as an input which are a mathematical representation of the words and often convey statistical information about e.g. word frequencies. Word2vec vectors are used in the tests because of their simplicity, but the algorithm is modular enough to allow any form of mathematical modelling for the word. An already pre-trained word model and a model specifically trained on the legal document corpus are tested in this thesis. The pre-trained model was trained on the German Wikipedia pages without any domain knowledge, which is probably the reason for the worse results.

The Adam algorithm is used for optimization. The learning rate is gradually decreased every five epochs by half, while the network was trained for 60 iterations or epochs over the whole training dataset. With this setup, the

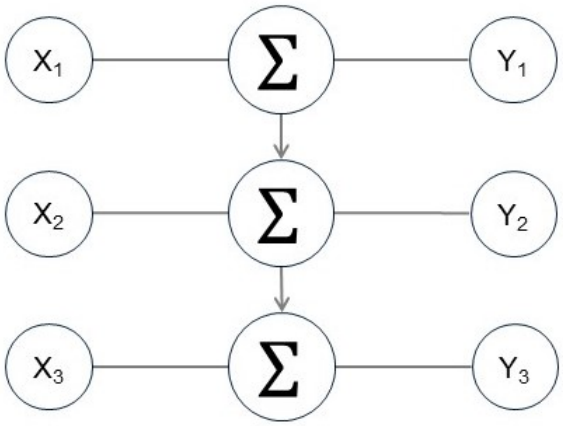


Figure 4.5.: RNN using information from 2 previous processing steps

different possible combinations are trained and tested, which are seen in section 5.4. The best model was determined and further refined.

4.4. Existing Approaches

SBD is an important process which can be found in almost any NLP framework. The tested, existing methods in this thesis are chosen based on the reported performance by [Re12]. OpenNLP had the best overall performance on the combined dataset without any preprocessing. The sentence segmentation module from NLTK is the only unsupervised approach with a performance levels comparable to the supervised approaches and a high robustness. Unfortunately the best reported method on all individual datasets "tokenizer" could not be found and consequently not tested, but the reported performance difference is only marginal. These methods will be used as a baseline for the other approaches.

4.4.1. Unsupervised: NLTK/Punkt

The Punkt module is part of the NLTK framework and one of the few existing unsupervised methods. The basic idea behind Punkt is to distinguish real sentence boundaries from abbreviations. The decision is based on the number of occurrences of a word followed by a dot versus the number of occurrences without a dot. When a word is always seen in combination with a dot, this is strong indicator for an abbreviation. Hypothesis testing is used to disambiguate abbreviations from sentence ends and based on the prediction the sentences are created as the output of the system [KS06]. Additionally it is possible to add a list of abbreviations to improve the system's performance[7].

This approach is the easiest to train of all tested methods, because only the text is needed as an input. It automatically scans the text for the occurrences of dots and then calculates the abbreviation probability. Generating the output predictions from the predicted sentences is more complex, because certain escape sequences are automatically removed from the text if they are placed at the end of a sentence. This results in a reduced token count, thus different predictions compared to the other methods. The missing tokens had to be readded.

Punkt has one crucial disadvantage: It can only detect abbreviations. Legal documents are more diverse in structure as described above with the different sentence types. Moreover Punkt cannot distinguish headlines. Thus in order to boost the performance of this module, it is also tested in combination with the template module.

4.4.2. Supervised: OpenNLP

OpenNLP is part of the Apache software projects[8] and uses trainable sentence segmenters. OpenNLP is a library for the Java programming language. This means the integration of this system into the module pool is more complex. Instead of writing a wrapper class for this system in Java, which would then be called from the Python code, the system was directly used via a system call.

[7] See NLTK [LB02] and its documentation
[8] https://opennlp.apache.org/

The system is trained on line-separated data meaning every sentence is on a new line. The output is in the same form, thus escape sequences are removed similar to Punkt. Those have to be automatically readded to produce the same token predictions. Because the system has potentially the ability to distinguish normal sentences from headlines or other structures, there is less motivation to use it in combination with the template module. During testing it even showed that those modules are also incompatible, producing really low test scores.

4.5. Methods Summary

The system introduced in this thesis combines different SBD methods and allows their application in the legal domain. OpenNLP and NLTK are previously existing, well established approaches and will be used as a baseline for the other methods. The rule-based and template approach try to utilize the structured format of legal documents. Based on the statistical properties CRF and RNN are able to predict the sentence boundaries. In the following chapter the performance of the different methods will be evaluated on German legal documents.

5. Performance Evaluation

As previously stated, the main focus of this evaluation is on the law and judgment collection. The other collections are used as a reference value because they do not consist of enough sentences to get reliable results for training and testing. After evaluating each method on those two collections, their performance is calculated on the other previously unseen datasets. The complete overview of the performance evaluation can be seen in appendix A.2.

To evaluate the NNs, the laws and judgments are each split into the three parts: train, validation and test dataset. The validation and test dataset each consist of 10% of the texts. Each different model is trained on the train dataset and after every epoch evaluated on the validation dataset. The model with the highest F1 score is saved and at the end of the training tested against the test dataset. The reported performance is always based on the test dataset to avoid implicit overfitting on the validation set.

The CRF are tested in a similar way. The only difference is that validation set is used at the end instead of throughout the training. The CRF is based on the external library CRFSuite, thus there are less possibilities of diverse testing schemes.

Each module is tested on the available data collections. For the dataset the module was trained on, the performance level during training is reported and the documents used for training stated in italics. OpenNLP can be considered overfit on those datasets, because it is hardly possible to produce the same test split used for the other methods, thus the method is tested on all documents. In order to understand the usefulness of the template method, each other approach is tested in combination with this method. The overall performance of all methods can be see in table 5.1

Performance is measured in F1 score, recall and precision because all other evaluation metrics such as accuracy are heavily influenced by the large num-

ber of negative examples for sentence boundaries. For example, an accuracy value of approximately 98% can be achieved by not assigning any sentence boundaries at all.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{Positive}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Precision indicates how many of the predicted sentence boundaries are actually sentence boundaries, recall calculates the fraction of sentence boundaries found and F1 can be seen as an average between the former. In this case a high recall score is more important than precision, because finer segmentation is not as grave as missing sentence boundaries as discussed in section 2.3. Nonetheless, the performance evaluation mostly focuses on the F1 score as it is encapsulates both aspects and is most commonly used in such a classification setting. In table 5.1 all the performance scores calculated in the following sections concerning laws and judgments are summarized.

5.1. Existing Approaches

The already existing approaches OpenNLP and NLTK perform the worst in comparison, with the OpenNLP model trained on judgments being slightly better (see table 5.2). An obscure fact to consider is that the OpenNLP model trained on judgments outperforms the model trained on laws when evaluated on laws. A reason for this could be the higher complexity of the law texts, also observable by the overall better performance on the judgments for almost all methods (comparing the results for each method specifically trained for that dataset). In this case, OpenNLP might not be capable of modelling the laws

	Laws			Judgments		
	F1	Rec	Pre	F1	Rec	Pre
Rule	82.7	74.7	92.8	88.6	88.3	88.9
NLTK	73.0	61.2	92.1	69.9	66.4	73.9
OpenNLP *Laws*	72.9	61.1	90.4	64.4	64.5	64.4
OpenNLP *Judgments*	74.8	60.3	**98.4**	78.1	65.6	96.3
CRF *Law*	95.4	94.4	96.4	77.3	68.4	89.0
CRF *Jug*	81.2	76.8	86.0	96.8	96.6	97.0
NN *Law*	**97.7**	**98.1**	97.3	84.3	80.9	88.0
NN *Jug*	87.5	84.4	90.9	**98.7**	**99.3**	**98.1**

Table 5.1.: Performance evaluation for every method on laws and judgments

adequately, but can generalize the concept of sentence boundaries well enough with the judgments.

	Laws			Judgments		
	F1	Rec	Pre	F1	Rec	Pre
NLTK	73.0	**61.2**	92.1	69.9	**66.4**	73.9
OpenNLP *Laws*	72.9	61.1	90.4	64.4	64.5	64.4
OpenNLP *Judgments*	**74.8**	60.3	**98.4**	**78.1**	65.6	**96.3**

Table 5.2.: Performance evaluation for NLTK and OpenNLP on laws and judgments

When visually examining the results obtained form this models it is evident that most errors are linked to structural textual units such as citations, headlines and lists. In order to overcome those expected problems the template module was created, which is also tested in combination with the other methods. As stated above, the template module encounters a problem when used in combination with the OpenNLP module, resulting in test scores far worse than without, thus it is not included here. There are more capable methods, therefore this problem is not further investigated.

In this case additional structural guidance increases performance of the Punkt system for laws, as seen in 5.3. The system gains more performance on the

	Laws			Judgments		
	F1	Rec	Pre	F1	Rec	Pre
Temp(NLTK)	**79.0**	**71.0**	89.2	69.1	**69.7**	68.5
NLTK	73.0	61.2	**92.1**	**69.9**	66.4	**73.9**

Table 5.3.: Performance of the template module in combination with NLTK

law dataset, which aligns with the fact that those include more structural information than judgments for which similar performance was achieved.

	ToS			Privacy Pol.		
	F1	Rec	Pre	F1	Rec	Pre
Temp(NLTK)	**84.8**	**85.2**	84.4	77.1	**73.1**	81.7
NLTK	80.0	77.0	83.2	76.9	73.0	81.3
OpenNLP *Laws*	79.5	76.6	82.6	76.7	71.5	82.9
OpenNLP *Judgments*	83.8	76.0	**93.3**	**80.2**	70.3	**93.2**

Table 5.4.: Performance of existing approaches on terms of service and privacy policies.

To summarize this performance evaluation of the existing solutions: for terms of service and privacy policies similar performance levels are achieved with results far from optimal as seen in 5.4. The advice is not to use those existing approach for sentence boundary detection in the German legal domain and instead utilize a tailored approach. Although, a method with a lot more preprocessing could also be possible, e.g. first extracting the structural information and then further process the text, similar to the template module.

The tested systems also seem to only predict a sentence boundary on a token they are certain on. This can be seen by the high precision score for the methods, with OpenNLP having the highest evaluated score. Of the boundaries predicted a high number is correctly classified, but the low recall score shows that only a limited set of boundaries is even considered.

With the existing approaches as the baseline, every other method achieves higher F1 scores, thus it is advised not to use them in the German legal domain. A small amount of labour is sufficient to produce better results.

5.2. Rule Module

Although only a short amount of time was used to create this module, it outperforms the existing approaches, see 5.5. The rule module was build with legal documents in mind, thus it might produce the most common errors concerning structural information. This assumption is further supported when visually examining the predictions via the created GUI.

	Laws			Judgments		
	F1	Rec	Pre	F1	Rec	Pre
Rule	**82.7**	**74.7**	92.8	**88.6**	**88.3**	88.9
Temp(NLTK)	79.0	71.0	89.2	69.1	69.7	68.5
OpenNLP *Judgments*	74.8	60.3	**98.4**	78.1	65.6	**96.3**

Table 5.5.: Performance evaluation of the rule module

The module has stable scores over all the datasets. Because of that this module outperforms the more sophisticated modules on the dataset they were not trained on. Overall, it is suggested to use this approach as a backup, if a system using sentence segmentation has variable performance. The overview, including the other datasets, can be seen in A.3.

With its limited rules, the rule module is not able to model all facets of legal documents. For example, abbreviations are never seen as sentence terminating. Nonetheless, stable performance levels are achieved.

5.3. Conditional Random Fields

Many different feature combinations for the CRF are tested and based on the average score of a feature extractor with its respective word window, the best ones are selected. Particularly widening the scope of a feature extractor leads to higher results, but increasing the window over a certain threshold often degrades the performance of that extractor. Using more lowercased words as a feature increases the model performance, but the computational and memory cost is to high to further increase the viewed scope. In the scope of this thesis

only symmetric feature windows, thus the same number of token before and after a given word, are used.

Using Conditional Random Fields lead to a huge performance boost in comparison to all the other methods discussed so far. On the dataset the CRF was trained on, the performance scores reached the higher 90%s. Only a handful of core features are essential for the high performance values of the model. Moreover, this allows for a faster training, less memory usage and faster predictions. In order to achieve more stable results the performance score of the CRF is averaged over at least three runs for the larger models with a maximum of 5 runs for the smaller models.

The best CRF trained on laws uses the following features: *Special* with a window of 10, *Lowercase* and *Length* with a window of 7, *Signature* with a window of 5 and *Lower*, *Upper* and *Number* with a window of 3.

The best CRF for the judgments has following extractors: *Length*, *Signature*, *Lowercase*, *Special* with a window of 5 and *Lower*, *Upper* and *Number* with a window of 3.

	Laws			Judgments		
	F1	Rec	Pre	F1	Rec	Pre
CRF *Law*	**95.4**	**94.4**	**96.4**	77.3	68.4	89.0
CRF *Jug*	81.2	76.8	86.0	**96.8**	**96.6**	**97.0**

Table 5.6.: Performance CRF on laws and judgments

Particularly the inclusion of special tokens is important for a good performance on the law documents. The assumption here is that they are essential for a better understanding about the overall structure. For training on judgments, a smaller scope for each extractor yields the best performance.

Overall the performance of CRF increases significantly with the inclusion of the *Lowercase* extractor. This extractor was introduced by [SA17b], but they used a smaller scope. A similar model to the one introduced in their paper[9] is also tested but produces worse results with an F1 score of 95.0% for laws and 96.0% for judgments.

[9]The only extractor not included was their binary value whether a token is found in the first part of a document

	ToS			Privacy Pol.		
	F1	Rec	Pre	F1	Rec	Pre
CRF *Law*	**91.8**	**86.4**	**95.7**	81.7	81.0	82.3
CRF *Jug*	81.1	82.4	79.8	**84.5**	**81.6**	**87.6**

Table 5.7.: Performance CRF on terms of service and privacy policies

The evaluation for the terms of service and privacy policies showed a challenge already seen in [Re12]. The performance of the SBD system significantly decreases when processing unknown documents, see table 5.7. The F1 score for the privacy policies nearly reaches the baseline value from the OpenNLP with of 84.5%.

CRF are a versatile model, but their performance levels are bound to the creativity while engineering the features. As seen during testing, changes to features often only have a marginal impact on the overall performance. Additionally, there are only a limited number of parameters possible. This is not the case with a RNN, which will be evaluated next.

5.4. Recurrent Neural Networks

Training the Neural Networks involves a lot of testing, because there are many more possible feature combinations than with for example the CRF. Thus certain ground parameters are fixed during the whole training. One of those parameters is the hidden dimension size of the recurrent unit in the networks. The aim of this evaluation is to show which areas have been tested and which maybe need more refinement.

One of the most important decisions is the way the text is processed by the NN. In this case, word embeddings are used as an input to the system with an additional context window provided before and after a potential sentence boundary. The window size is chosen in accordance to the best scopes with the CRF for more complex features. Seven tokens before and after a possible sentence boundary are used. As already stated, only bidirectional recurrent units are used to give models the ability to utilize previous and future token information.

Unit	Input	Laws			Judgments		
		F1	Rec	Pre	F1	Rec	Pre
RNN	word2vec	97.2	**97.8**	96.7	98.6	98.7	**98.4**
LSTM	word2vec	**97.4**	97.7	**97.1**	98.3	98.5	98.1
GRU	word2vec	**97.4**	98.2	96.6	**98.7**	**99.3**	98.1
RNN	pret. word2vec	86.6	82.5	91.1	89.7	87.5	92.1
LSTM	pret. word2vec	86.7	84.0	89.6	88.8	85.6	92.3
GRU	pret. word2vec	86.3	82.3	90.7	89.6	87.3	92.1

Table 5.8.: Performance comparison between different word embeddings

As the input to the RNN word2vec word embeddings are utilized, because of their simplicity. They are trained on the dataset collected for this thesis with a vector size of 100. Additionally, one set of pre-trained embeddings calculated from German Wikipedia articles with a vector size of 300[10] is used. With those embeddings simple recurrent NN are trained with RNN, LSTM and GRU as recurrent unit with their standard non-linear activation function (Tanh, Sigmoid), followed by two linear layers with a rectified linear unit function in between and a sigmoid as the output. The hidden dimension is 64, as well as the output of the first linear unit.

If a word is not in the embedding dictionary, its word vector is set to zero. Other strategies exist as well, for example averaging the embeddings in a window to estimate the word vector, but bidirectional recurrent layers are specifically created to also model the context of a word.

As seen in table 5.8, it does make a difference which embeddings to use, and in this case the difference is quite big. Based on those values, the decision for the further testing the recurrent unit is made. The LSTM and the GRU models have the highest F1 scores on the two datasets, so those recurrent units are used for further testing.

The next tests focus on the architecture of the RNN. Based on those results the best architecture is chosen. The features to vary are the number of recurrent layers at the beginning of the network and the further processing. In this thesis a combination of convolutional layers and linear layers is also tested. MaxPooling or the rectified linear unit are used as non-linearities.

[10] https://devmount.github.io/GermanWordEmbeddings/

Architecture	Laws			Judgments		
	F1	Rec	Pre	F1	Rec	Pre
2 *rec*, 1 *lin*	96.0	96.6	95.5	96.6	96.7	97.0
2 *rec*, 2 *lin*	97.4	97.7	97.1	*98.3*	**98.5**	98.1
2 *rec*, 4 *lin*	**97.7**	**98.1**	**97.3**	*98.3*	98.8	97.8
4 *rec*, 1 *lin*	96.0	96.1	95.9	96.2	96.0	96.4
4 *rec*, 2 *lin*	97.5	**98.1**	97.0	*98.1*	98.6	97.7
4 *rec*, 4 *lin*	96.5	96.3	96.7	97.9	98.0	97.9
8 *rec*, 2 *lin*	96.8	97.7	95.9	**98.3**	98.3	**98.3**
2 *rec*, 2 *conv*, 2 *lin*	96.7	97.1	96.3	97.8	98.0	97.3

Table 5.9.: Performance of different LSTM models

Architecture	Laws			Judgments		
	F1	Rec	Pre	F1	Rec	Pre
2 *rec*, 1 *lin*	95.8	95.9	95.7	96.4	96.5	96.2
2 *rec*, 2 *lin*	97.4	**98.2**	96.6	**98.7**	**99.3**	98.1
2 *rec*, 4 *lin*	97.5	**98.2**	96.8	98.1	99.0	97.2
4 *rec*, 1 *lin*	95.6	96.5	94.6	96.3	95.7	96.9
4 *rec*, 2 *lin*	97.1	97.6	96.6	**98.7**	98.9	**98.5**
4 *rec*, 4 *lin*	**97.6**	98.0	**97.2**	98.3	98.8	97.8
8 *rec*, 2 *lin*	97.3	97.6	97.0	98.1	98.5	97.7
2 *rec*, 2 *conv*, 2 *lin*	96.5	96.4	96.6	97.8	97.3	98.3

Table 5.10.: Performance of different GRU models

Convolutional layers do not add any value in this context, see table 5.9 and 5.10, although those networks possibly have more modelling capabilities with their bigger structures. The RNN with a GRU unit with four recurrent and two linear layers outperforms all other methods on judgments with an F1 score of 98.7%. With four recurrent and four linear layers, the RNN achieves a F1 score of 97.6%, thus only slightly worse results than the LSTM model with two recurrent and four linear layers and a score of 97.7%.

Similarly to the other methods, there is a performance problem, when evaluating the best models on the other datasets besides laws and judgments, see 5.11, although the NN still achieves the best results on the terms of service. The NN performs worse than the CRF on privacy policies. The NN is not able to generalize on other documents and the hypothesis is that the word embeddings are not capable of modelling the statistical properties of those texts well.

	ToS			Privacy Pol.		
	F1	Rec	Pre	F1	Rec	Pre
NN *Law*	91.3	**89.9**	92.7	**82.6**	**84.0**	81.3
NN *Jug*	**92.3**	87.3	**97.7**	81.2	77.0	**85.8**
NN pre. *Law*	57.2	49.1	68.4	33.7	33.1	34.3
NN pre. *Jug*	55.1	51.9	58.8	40.1	36.4	44.5

Table 5.11.: Performance recurrent NN on terms of service and privacy policies

In contrast to the laws and judgments, terms of service and privacy policies are relatively small. Only with a reasonable large document corpus statistically significant embeddings can be build. The pre-trained word embeddings produce even worse results.

Another disadvantage of this approach is the time it takes to produce all the predictions for a given texts. The NN requires the most processing time out of all the approaches tested in this thesis. When dealing with a large number of legal documents, this is one fact to keep in mind.

5.5. Wikipedia

Although high performance scores are achieved for legal documents, a performance decline can be observed, when not evaluating a model on documents it was not trained on. In order to evaluate the performance of the modules created for this thesis further, they are tested on German Wikipedia articles to emulate normal text. This evaluation can be used as an indicator to assess the stability of the systems. The testing focusses on the best modules created of the NNs and CRF on their respective dataset denoted in italics.

	Wikipedia		
	F1	Rec	Pre
CRF *Law*	83.8	76.7	92.4
CRF *Jug*	82.7	76.7	**89.8**
NN *Law*	**87.3**	**86.8**	87.7
NN *Jug*	83.2	78.8	88.1

Table 5.12.: Performance evaluation for NN and CRF on German Wikipedia articles

As seen in 5.12, the created methods have a performance issue in this case with a decline of around 10%. The assumption is that the modules are to specific to generalise well on other domains, although the performance difference between the source and target domain is bigger than reported in other research on other domains, see [Re12].

5.6. XML

The last evaluation deals with the hypothesis that structured documents are easier to segment into sentences. NNs generally needs a lot of data to train and already achieved high performance scores, thus a CRF was trained to evaluate the usefulness of more structural information. The performance on the annotated XML files of the GG and SGB 1 can directly be compared to their plaintext versions.

The performance on laws in table 5.13 is directly taken from the evaluation routine found above, thus a comparision cannot be done directly. Evaluating the best CRF for laws on those two documents would only produce an overfitted performance score. In this case the XML dataset consists of less documents than the law collection, resulting in less variability between the given documents. The performance scores a based on all the documents in the dataset, whereas in the case of the XML documents only the GG and SGB 1 are tested. The best feature combination for the CRF on laws is used here and trained with 30% of the documents as a test dataset. The larger train-test split is due to the smaller XML dataset, to still achieve statistically significant results.

	Law			XML		
	F1	Rec	Pre	F1	Rec	Pre
CRF *Law*	**95.4**	**94.4**	**96.4**	85.6	77.4	95.7
CRF *XML*	81.4	73.7	90.9	**97.0**	**97.5**	**96.4**

Table 5.13.: Performance evaluation for CRF on XML

Table 5.13 indicates a performance increase with using additionally annotated documents. For a definitive answer more data needs to be collected with a

wider variety. Using documents with markup information seems to improve the performance achievable by a SBD system, but has two disadvantages.

First, XML documents store less sentences for the same amount of textual data. As seen in table 3.1, the XML collection consists of less sentences than the privacy policies or terms of service, but has almost two times the number of tokens of those datasets. During the annotation process this was also observed, with a significantly increased labour effort needed when processing the XML documents.

Second, the XML documents are most of the time created manually, as discussed in 6.2. This leads to the issue that manual labour is needed to reliably segment documents into sentences which in turn shall be used to simplifiy manual labour. In this case a sentence segmentation system operating on raw text is more preferable, which could be used to create XML documents or assist in the creation process.

5.7. Summary

Throughout this chapter many different methods are tested, each achieving different performance levels. The previously existing methods do not work well on legal documents with their more complex structures and it is not advised to use them in the legal domain. Using additional structural knowledge with a preprocessing step such as the Template module, can help increase their performance, but this is no optimal solution. The CRF and NN achieve significantly better results with an F1 score of almost 99% for one RNN consisting of GRU recurrent units. Still, the problem of stability remains, as their performance decreases on other datasets in some cases.

The errors made by CRF and NN are hard to describe, because they often consist of a very specific combination of special characters and alphanumeric tokens. For example, there is sometimes a headline which is not annotated and with the only difference to other headlines being the usage of numbers or special characters before it. For such errors, further features might be needed, although they are very rarely found.

Highly specialized models are crucial to ensure an overall good performance in the legal domain. Further variations are possible to increase model robustness against dataset variants and will be discussed in the next chapter concerning future work.

6. Future Work

6.1. Scientific Work

Although, a lot of evaluation on SBD systems was done in this thesis and excellent results were achieved, there are still some challenges and research paths to explore.

There are several question concerning model performance. The NN has a lot more possibilities that can be tested in the future, e.g. a study on different word embeddings as an input to the system would be really interesting to see. The question for future work is whether sophisticated word embeddings for the neural network would improve the results. Word2vec is a fairly simple model and other models like BERT or ELMo additionally model the context of a word. Another possibility to investigate is the combination of a neural network and a CRF, where the NN calculates the feature inputs. Such a model was also used in other domains.

In this case one further question is, how to increase the stability of the model in order to make it more usable in the general legal domain. Additionally one model for each individual document type handled by a system can be created. In this case the decision of the model still needs to be clarified, because it is no trivial task to decide on the model best describing the current document. A possible answer would be to determine the similarity of different documents and then chose the model trained on the most similar document type. Otherwise there might be performance problems with a given system, induced by using an unfitting model. Another possibility is to label a small number of sentence boundaries in the documents and then choosing the best model based on a performance evaluation.

Another issue not further investigated in this thesis is, that the nature of the task leads to a huge class imbalance between sentence boundary tokens and

non-sentence boundary. This challenge is not investigated further because the results already showed high performance for the used methods. The segmentation result might improve when investigating solutions to this issue.

From a scientific point of view, there are some experiments and questions open to create a reliable and versatile SBD system for the German legal domain, but the state-of-the-art results already allow a practical application of this work.

6.2. Practical Application

In order to further stress the need for performant NLP tools, one challenge of a publisher in the legal domain will be addressed in the following paragraphs. Publishers have a special role in the legal domain, because they keep and enrich the information produced every day, thus help legal practitioners with their work.

The *Dr. Otto Schmidt* publisher is facing some challenges while processing legal documents. To stress the importance, the process of incorporating new documents into their database will be briefly described. Books or other texts are written in text processing application like Word. This document is then most of the time segmented and translated manually via a type system to an XML or DTD document, which is richly annotated. Those documents are used to automatically extract links and segment the text into the individual paragraphs later on. The structured format is necessary to display and link different documents. Every document is catalogued. By using a rich regular expression rule set the individual links are extracted, that means external links (to other documents) or internal links (within the text, e.g. to notes) are identified.

Most of the processing work is done by manual labour, but there is an attempt to automatize this work at least for older documents. Now there is some work done to automatically extract some information from judgments in collaboration with this chair. Since the structure is mostly similar here, an automatic system could potentially achieve high performance, if it is tailored to those document types.

A direct application for a SBD system exists in this case. A parser can use the individual sentences and assigned them to certain sub-parts of the text needed such as paragraphs, lists or extract information. Overall, a mapping between the type system used and the segments produced by the SBD system is required. The SBD system would allow to automatically partition the text into the parts needed.

Additional labour is needed, but the individual sentences formulate a ground-truth usable by NLP in the German legal domain.

7. Conclusion

The aim of the thesis was to investigate SBD in German legal documents. Sentences in the legal domain are more complex to distinguish with the different structural components inherent to the texts. They need to be reliable identified to segment the sentences and partition them from any unwanted pieces of information.

Based on the definition for sentences in the German legal domain, a dataset was collected to evaluate the performance of multiple existing and newly created methods. A wide variety of different document types was collected for an overview of SBD performance in the German legal domain.

The nearly perfect performance levels reported in the related work can hardly be reproduced in the legal domain, possibly caused by the more complex structure of legal documents. Additionally, an evolving definition for SBD can be observed from abbreviation disambiguation to a logical segmentation of documents. Sentences are more complex, which is especially evident in the legal domain.

A system was implemented with a high modularity and a diverse portfolio of methods. Evaluating those approaches showed issues with existing methods in the legal domain. A significant amount of structural knowledge is needed to successfully segment sentences. When tailored to the legal domain, methods found in related research such as CRF and NN are able to achieve state-of-the-art results.

When comparing the performance of the created system on different document types it is inherent, that additional models might be needed. The models trained in this thesis are not able to generalise the concept of SBD well over all document types.

SBD is only one part for many NLP applications and still remains a challenge often overseen and still not 100% solved. NLP methods should aim for

broad use cases, but the effort spend to build tailored systems should be waged against the possible performance increases. Particularly the legal domain introduces new challenges with its more complex structures, which are often not solvable in a generic way.

The groundwork done in this thesis should support further research and applications, which try to extract information from documents found in the legal domain. It is important to get a good understanding about the possible performance of individual system parts and modules. The introduced SBD system tailored for the legal domain can easily be expanded and further refined.

Computer systems are shaping work in many different domains, but legal practitioners are very critical with changes to their working routine. Particularly, intelligent systems are a matter often debated. Nonetheless, it will be interesting to see how technology can shape the legal domain in the future.

Bibliography

[Ab95] Aberdeen, J.; Burger, J.; Day, D.; Hirschman, L.; Robinson, P.; Vilain, M.: *MITRE: description of the Alembic system used for MUC-6.* In *Proceedings of the Sixth Message Understanding Conference (MUC-6).* 11 1995. 4

[De12] Dell'Orletta, F.; Marchi, S.; Montemagni, S.; Plank, B.; Venturi, G.: *The SPLeT-2012 Shared Task on Dependency Parsing of Legal Texts.* In *Proceedings of the 4th Workshop on Semantic Processing of Legal Texts.* Istanbul, Turkey. 2012. 2.1

[Gi09] Gillick, D.: *Sentence Boundary Detection and the Problem with the U.S.* In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers.* pages 241–244. Boulder, Colorado. June 2009. Association for Computational Linguistics. 3, 4.1

[Gr15] Grabmair, M.; Ashley, K. D.; Chen, R.; Sureshkumar, P.; Wang, C.; Nyberg, E.; Walker, V. R.: *Introducing LUIMA: An Experiment in Legal Conceptual Retrieval of Vaccine Injury Decisions Using a UIMA Type System and Tools.* In *Proceedings of the 15th International Conference on Artificial Intelligence and Law.* ICAIL '15. pages 69–78. New York, NY, USA. 2015. ACM. 1.1

[GWM18] Glaser, I.; Waltl, B.; Matthes, F.: *Named Entity Recognition, Extraction, and Linking in German Legal Contracts.* In *Proceedings of IRIS: Internationales Rechtsinformatik Symposium.* IRIS. Salzburg, Austria. 2018. 1

[JW13] Jurish, B.; Würzner, K.-M.: *Word and Sentence Tokenization with Hidden Markov Models. JLCL.* 28:61–83. January 2013. 4, 4.1

[Ko19] Koller, D.: *Interactive Analysis of a Corpus of General Terms and Conditions for Variability Modeling.* Master's thesis. Technical University of Munich. Munich, Germany. April 2019. 3

[KS02a] Kiss, T.; Strunk, J.: *Scaled Log Likelihood Ratios for the Detection of Abbreviations in Text Corpora.* In (Tseng, S.-C., Ed.): *Proceedings of COLING 2002.* pages 1228–1232. Taipeh. 2002. 4, 4.1

[KS02b] Kiss, T.; Strunk, J.: *Viewing sentence boundary detection as collocation identification.* In (Busemann, S., Ed.): *Konvens 2002 Tagungsband.* pages 75–82. Saarbrücken, Germany. 2002. 4.1

[KS06] Kiss, T.; Strunk, J.: *Unsupervised Multilingual Sentence Boundary Detection. Comput. Linguist.* 32(4):485–525. 12 2006. 4.1, 4.4.1

[LB02] Loper, E.; Bird, S.: *NLTK: The Natural Language Toolkit.* In *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics.* Philadelphia. July 2002. Association for Computational Linguistics. 7

[LMP01] Lafferty, J. D.; McCallum, A.; Pereira, F. C. N.: *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data.* In *Proceedings of the Eighteenth International Conference on Machine Learning.* ICML '01. pages 282–289. San Francisco, CA, USA. 2001. Morgan Kaufmann Publishers Inc. 4, 4.3.3

[Ma12] de Maat, E.: *Making sense of legal texts.* PhD thesis. University of Amsterdam. 2012. 1.1, 2.3

[Mi00] Mikheev, A.: *Tagging Sentence Boundaries.* In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference.* NAACL 2000. pages 264–271. Stroudsburg, PA, USA. 2000. Association for Computational Linguistics. 2, 4.1

[Mi02] Mikheev, A.: *Periods, Capitalized Words, etc.* In *Comput. Linguist.* volume 28(3). pages 289–318. Cambridge, MA, USA. September 2002. MIT Press. 4.1

[Mo04] Moens, M.: *Innovative techniques for legal text retrieval.* In *Artificial Intelligence and Law.* volume 9(1). pages 29–57. Netherlands. 2004. Kluwer Academic Publishers. 1

[MW09] de Maat, E.; Winkels, R.: *A Next Step Towards Automated Modelling of Sources of Law.* In *Proceedings of the 12th International Conference on Artificial Intelligence and Law.* ICAIL '09. pages 31–39. New York, NY, USA. 2009. ACM. 1.1

[MW10] de Maat, E.; Winkels, R.: *Automated Classification of Norms in Sources of Law.* In (Francesconi, E.; Montemagni, S.; Peters, W.; Tiscornia, D., Ed.): *Semantic Processing of Legal Texts.* pages 170–191. Springer-Verlag. Berlin, Heidelberg. 2010. 1.1

[Nu90] Nunberg, G.: *The Linguistics of Punctuation.* Center for The Study of Language (CSLI). 1990. 2.1

[Ok07] Okazaki, N.: *CRFsuite: a fast implementation of Conditional Random Fields (CRFs).* 2007. 4.3.3

[PH97] Palmer, D. D.; Hearst, M. A.: *Adaptive Multilingual Sentence Boundary Disambiguation. Comput. Linguist.* 23(2):241–267. June 1997. 4, 4.1, 2

[Re12] Read, J.; Dridan, R.; Oepen, S.; Solberg, L. J.: *Sentence Boundary Detection: A Long Solved Problem?* In *Proceedings of COLING 2012: Posters.* pages 985–994. Mumbai, India. December 2012. The COLING 2012 Organizing Committee. 1.1, 3, 4, 4.4, 5.3, 5.5

[RR97] Reynar, J. C.; Ratnaparkhi, A.: *A Maximum Entropy Approach to Identifying Sentence Boundaries.* In *Proceedings of the Fifth Conference on Applied Natural Language Processing.* ANLC '97. pages 16–19. Stroudsburg, PA, USA. 1997. Association for Computational Linguistics. 2, 4, 4.1

[SA17a] Savelka, J.; Ashley, K. D.: *Sentence Boundary Detection in Adjudicatory Decisions in the United States. Traitement automatique des langues.* 58(February):21–45. 2017. 2.1, 2.3, 4.1.1

[SA17b] Savelka, J.; Ashley, K. D.: *Using Conditional Random Fields to Detect Different Functional Types of Content in Decisions of United*

States Courts with Example Application to Sentence Boundary Detection. In *Proceedings of 2nd Workshop on Automated Semantic Analysis of Information in Legal Texts.* London, Great Britain. June 2017. ASAIL. 2.1, 4.1.1, 4.2.3, 5.3

[Sc00] Schmid, H.: *Unsupervised Learning of Period Disambiguation for Tokenisation.* In *Internal Report.* University of Stuttgart. May 2000. 4.1

[SFK99] Stamatatos, E.; Fakotakis, N.; Kokkinakis, G.: *Automatic Extraction of Rules for Sentence Boundary Disambiguation.* In *Workshop on Machine Learning in Human Language Technology.* Chania, Greece. 1999. 4

[Sh18] Shao, Y.: *Segmenting and Tagging Text with Neural Networks.* PhD thesis. Uppsala Universitet. Uppsala, Sweden. 2018. 4

[SM11] Sutton, C.; McCallum, A.: *An Introduction to Conditional Random Fields.* In *Foundations and Trends in Machine Learning.* volume 4(4). pages 267–373. November 2011. 4.3.3

[Su18] Sugisaki, K.: *Word and Sentence Segmentation in German: Overcoming Idiosyncrasies in the Use of Punctuation in Private Communication.* In (Rehm, G.; Declerck, T., Ed.): *Language Technologies for the Challenges of the Digital Age.* volume 10713 of *Lecture Notes in Computer Science.* Springer, Cham. 2018. 4, 4.1

[Wa17] Waltl, B.; Muhr, J.; Glaser, I.; Scepankova, E.; Bonczek, G.; Matthes, F.: *Classifying Legal Norms with Active Machine Learning.* In *Proceedings of Jurix: International Conference on Legal Knowledge and Information Systems.* Jurix. Luxembourg. 2017. 1

[WP10] Wyner, A.; Peters, W.: *Towards annotating and extracting textual legal case factors.* In *Proceedings of the Language Resources and Evaluation Conference Workshop on Semantic Processing of Legal Texts.* 2010. 1

[WP11] Wyner, A.; Peters, W.: *On Rule Extraction from Regulations.* In *JURIX.* volume 11. pages 113–122. 2011. 1

A. Appendix

A.1. Document Statistics

Table A.1.: BGB Paragraphs

Identical sentence start *Num.:* **145**	§55a, §79, §104, §126b, §199(1), §199(3), §212, §271a(2), §286, §305a, §307, §312a, §312b, §312f(3), §312i, §323, §346, §356(2), §356(5), §357, §357a, §376, §434, §437, §438, §489, §491, §492b, §493(4), §495, §498, §502(2), §505c, §505d, §505e, §506, §510, §536a, §536c, §542, §543, §545, §555b, §556c, §556d, §556g, §558, §559, §559b, §559c, §559d, §573, §573b, §576a, §577a, §580a, §589, §595(3), §605, §621, §622, §630e, §633, §634, §634a, §641, §650f, §650g, §651a(2,4,5), §651b, §651c, §651f, §651g, §651i, §651k(1), §651n, §651o, §651p(1), §651r(1), §651t, §651u(2), §651w, §651x, §675d, §675e, §675i, §675k, §675m, §675q, §675s, §657t, §675v, §675x, §676a, §676b, §676c, §701, §702, §723, §773, §775, §1179, §1310(1), §1314, §1315, §1316, §1318, §1375, §1379, §1385, §1390, §1418, §1447, §1455, §1469, §1495, §1561, §1579, §1592, §1597a, §1613, §1617c, §1626a, §1629, §1671, §1747, §1757, §1760, §1762, §1763, §1772, §1778, §1781, §1786, §1813, §1817, §1905, §1906, §1906a, §1908f, §2060, §2109, §2163, §2333, §2339

Variable sentence middle *Num.:* **17**		§207(1), §312h, §350, §438, §505a, §575, §595(1), §650b, §655a, §675w, §1310(3), §1568a, §1571, §1572, §1598a, §1612a, §1836d
Enumerations	Standard *Num.:* **42**	§58, §75, §81, §101, §197, §204, §207, §271a(5), §311, §312, §312f, §312g, §479, §493(5), §549, §555c, §555f, §558a, §613a, §651a(3), §651b(2) §651h(2), §651k(5), §651p(3), §651q, §651r(2), §651u(3), §1441, §1463, §1478, §1499, §1600, §1609, §1666, §1686a, §1712, §1795, §1807, §1821, §1822, §1903, §2231
	Sentences Num.: **9**	§98, §310, §502(3), §576, §569, §651h(4), §1059a, §1612b, §1836c
Definition *Num.:* **2**		§308, §309
Lists of lists *Num.:* **18**		§204, §207, §305a, §308, §309, §312, §356, §429b, §438, §498, §543, §651a, §651f, §651h, §651k(5), §651w, §1315, §1613
Footnotes & Official note *Num.:* **63**		§14, §241a, §247, §271a, §275, §286, §288, §304, §308, §310, §311b, §313, §314, §323, §326, §345, §432, §487, §489, §490, §556d, §556e, §556f, §556g, §557, §557a, §557b, §558, §559, §559a, §559b, §559c §559d, §561, §568, §569, §573, §573a, §573b, §573c, §573d, §575, §575a, §577, §577a, §578, §610, §613a, §650, §674, §723, §724, §725, §727, §728, §1357, §1376, §1565, §1566, §1573, §1617, §1754, §1755

Table A.2.: Documents in the Dataset

Document Type	Documents
Laws	BGB, GG, SGB 1, SGB 2, SGB 3, StGB
Judgments	1 AR 15/19, 1 AR 16/19, 1 AR 21/19, 1 AR 30/19, 1 BvQ 36/19, 1 BvQ 42/19, 1 BvQ 43/19, 1 BvQ 45/19, 1 BvQ 46/19, 1 BvR 30/19, 1 SHa 22/18, 2 BvQ 45/18, 2 BvR 1728/16, 2 BvR 2425/18, 2 CE 19.515, 2 K 524/18, 2 Sa 341/18, 2 Sa 402/18, 2 Ta 10/19, 2 Ta 107/18, 2 Ta 142/18, 3 Ca 615/18, 3 K 1976/17, 3 U 22/19, 3 W 16/19, 4 HK = 14312/18, 4 K 268/17, 4 K 1057/18, 4 Sa 81/18, 4 Sa 306/18, 4 Sa 336/18, 4 Sa 514/18, 4 Ta 120/18, 4 Ta 133/18, 4 Ta 136/18, 4 TaBV 19/18, 5 K 887/18, 5 K 1199/17, 5 Sa 100/18, 5 Sa 153/16, 5 Sa 222/17, 5 Ta 104/18, 5 Ta 144/18, 5 TaBV 53/18, 6 K 130/18, 6 K 401/18, 6 K 424/17, 6 K 508/18, 6 K 719/18, 6 K 1373/18, 6 K 3063/18, 7 CE 18.2023, 7 K 410/15, 7 K 484/17, 7 K 1302/18, 7 K 3219/18, 7 O 2141/17, 7 Sa 95/18, 7 Sa 206/18, 7 Sa 256/18, 7 Sa 365/18, 7 U 2711/18, 7 V 2273/18, 7 V 2652/18, 8 K 142/17, 8 m 19.30137, 8 Sa 176/18, 8 Sa 250/18, 8 Sa 348/18, 8 Sa 348/18, 8 ZB 17/573, 9 C 18.2676, 9 ZB 16/2615, 10 K 1883/17, 11 C 19.632, 11 ZB 19.213, 12 K 23/19, 12 K 715/17, 12 K 1315/18, 12 K 1888/18, 13 U 1296/17, 14 K 1039/16, 14 K 1825/17, 14 S 15269/18, 15 K 1181/18, 15 V 2627/18, 20 U 124/19, 20 U 4223/18,

	23 U 2693/18, 31 Wx 194/19, 31 Wx 216/19, 31 Wx 216/19 Kost, 31 Wx 428 18, 36 Ca 11585/17, 130 C 60/17, 202 ObOWi 460/19, 241 C 24131/18, AN 17 X 18.01897, Au 1 K 18.1329, Au 5 K 18.1428, Au 5 K 18.31137, Au 8 K 18.1467, IX B 60/18, L 3 SF 160/18 AB, L 4 P 67/17, L 7 U 396/16, L 10 Al 23/19 B ER, L 11 AS 335/18, L 19 R 622/18, L 20 KR 262/17, M 25 S 18.3751, M 27 S 19.31719, S 11 AY 38/19, S 38 KA 5001/15, S 46 AS 785/19 ER, Verg 3/19, Verg 13/18, Vf. 1-VII-17, Vf. 1-VII-18, Vf. 10-VI-17, Vf. 17-VII-17, Vf. 20-VII-17, Vf. 21-VII-17, Vf. 23-VI-16, Vf. 29-VI-18, Vf. 32-VI-17, Vf. 39-VI-18, Vf. 60-VI-17, Vf. 66-VI-17, Vf. 74-VI-17, Vf. 81-VI-17, W 8 S 19.443
Terms of Service	21run, 0815 online, ACV Computerservice, AFA-Fotohandel, Agire, Ambiente AMS E-Commerce, ARDMED, ARS24, ati, BaSBa, batterium, BRASTY, bueroshop24, celexon, Chal-Tec, Cocopanda.de, Computeruniverse, comtech, CSC Spretz, CSMusiksysteme, cw-mobile, Dirk & Thomas Wermuth, Drogerie-Depot, drucker-guenstig, Durmeier Zaffran, Easynotebooks.de, eCom Dresden, Eichhorn, ELEKTRO-PLUS, elektroradar, ellerbrock, expert, eXXpozed, Eye & Lenses, f1-reifen, FC-Moto, Berlet, Flaconi, Foto Erhardt, fotokoester, Fox Trading, Gamingoase.de, fritzreifen, H. von Roon, Harald Hamp, hm-sat, HQ-Patronen, insani, Internetstores Bruegelmann, Internetstores Fahrrad, ipc-computer, Jacob Elektronik, Kochforum, marsmedia, Media Markt, Milan, MP-Vertriebs, mp3-players, MSK Bürotechnik, MVI Media, myToolStore, myToys, notebooksbilliger.de, Office Partner, Outlet46.de, Parfümerie Akzente, PARFUMERIE discounter, ParfumGroup, PDAMAX, Pharmeo, Pixxass, point-rouge, Premiumshop24, Priz, real, RedZilla, Reifendachs,

	Rudolf Wiegand und Partner, SADIK AKSOY, SatKing, Satkontor, Saturn, SGRO, Smartino, Sport-Tiedje, Technik-Profis, Ten Dance Media, TintenCenter.com, Tirendo, Toms Car HiFi, ukw24, Apotheke DocMorris, voelkner, Wagner eCommerce, Warsteiner Fotoversand, Zahrt
Privacy Policy	Accenture, BMW, Facebook, Google, IBM, Microsoft, Netflix, SAP, Sony, Amazon, Samsung
Wikipedia	Bürger, Data Mining, Interdisziplinarität, Klient, Legal Tech, Maschinelles Lernen, Online Rechtsberatung, Rechtsberatung, Rechtsdienstleistung, Recthsinformatik, Software, Staat, Wissen
XML	GG, SGB 1

A.2. Performance Evaluation

	Laws			Judgments			ToS			Privacy Pol.		
	F1	Rec	Pre	F1	Rec	Pre	F1	Rec	Pre	F1	Rec	Pre
Rule	82.7	74.7	92.8	88.6	88.3	88.9	86.3	83.5	89.1	83.3	81.2	85.5
Temp(Rule)	72.0	73.7	70.3	85.4	88.0	82.9	71.9	81.0	64.6	82.9	81.2	84.6
Temp(NLTK)	79.0	71.0	89.2	69.1	69.7	68.5	84.8	85.2	84.4	77.1	73.1	81.7
NLTK	73.0	61.2	92.1	69.9	66.4	73.9	80.0	77.0	83.2	76.9	73.0	81.3
OpenNLP *Laws*	72.9	61.1	90.4	64.4	64.5	64.4	79.5	76.6	82.6	76.7	71.5	82.9
OpenNLP *Judgments*	74.8	60.3	**98.4**	78.1	65.6	96.3	83.8	76.0	93.3	80.2	70.3	**93.2**
CRF *Law*	95.4	94.4	96.4	77.3	68.4	89.0	91.8	86.4	95.7	81.7	81.0	82.3
CRF *Jug*	81.2	76.8	86.0	96.8	96.6	97.0	81.1	82.4	79.8	**84.5**	81.6	87.6
NN *Law*	**97.7**	**98.1**	97.3	84.3	80.9	88.0	91.3	**89.9**	92.7	82.6	**84.0**	81.3
NN *Jug*	87.5	84.4	90.9	**98.7**	**99.3**	**98.1**	**92.3**	87.3	**97.7**	81.2	77.0	85.8

Table A.3.: Performance evaluation on the whole legal dataset

The performance evaluation is used to compare the different produced models. The information in italics denote the dataset the model was trained on. The evaluation calculated during training is used for the datasets the model was trained on.