Thesis for the degree of Licentiate of Engineering

Enhancing glass-box methods for part-of-speech tagging and text classification

MINERVA SUVANTO

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2025

Enhancing glass-box methods for part-of-speech tagging and text classification
Minerva Suvanto

Thesis for Licentiate of Engineering

Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE–412 96 Göteborg
Sweden
Telephone: +46 (0)31–772 1000

Chalmers digitaltryck
Göteborg, Sweden 2025

Enhancing glass-box methods for part-of-speech tagging and text classification

Minerva Suvanto
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology

Abstract

This thesis explores interpretable methods in two natural language processing (NLP) tasks, namely part-of-speech (POS) tagging and text classification. Currently, the NLP field is centered on the development and deployment of deep neural networks (DNNs), which have been established as the state-of-the-art and exhibit high performance in a variety of benchmarks. These models are to all intents and purposes black-boxes that lack interpretability, which is a major disadvantage when considering their use in high-stakes situations where transparency is essential.

The focus of this thesis is therefore on enhancing inherently transparent (glass-box) methods, thus establishing a foundation for their use in high-stakes scenarios. First, the task of POS tagging is considered. While often considered a solved problem, the findings here show that several challenges in POS tagging still remain. Based on these findings, a rule-based approach is explored for correcting the output of POS taggers. Second, interpretable text classification is studied with an enhanced linear classification method. The results demonstrate that a fully interpretable classifier can achieve a high performance when using the proposed enhancements, approaching that of pretrained DNN-based methods.

Keywords: part-of-speech tagging, text classification, natural language processing, glass-box methods, interpretability

For out of olde feldes, as men sey,
Cometh al this newe corn fro yeer to yere;
And out of olde bokes, in good fey,
Cometh al this newe science that men lere.

— Geoffrey Chaucer, Parlement of Foules, ll. 22–25

Acknowledgements

First and foremost, I want to express my gratitude to my main supervisor Prof. Mattias Wahde for the opportunity to pursue my academic career in an area that I find sincerely meaningful. Your advice has been absolutely essential in the creation of this thesis. I also want to extend my gratitude to my co-supervisor, Dr. Marco L. Della Vedova. I eagerly look forward to continuing my research journey in collaboration with both of you.

I am thankful to all of my colleagues in the AAI group as well as everyone at VEAS, in particular all my fellow PhD students going through the same journey. It is a pleasure to be part of the welcoming work environment that you have created here.

Thank you to my family and friends, in and outside of Gothenburg, and my dear partner for the endless support and encouragement that you give me. I am lucky to have you be part of my life.

List of included papers

Paper A Wahde, M., Suvanto M., and Della Vedova, M. L., *A Challenging Data Set for Evaluating Part-of-Speech Taggers*. In: Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART, pp. 79-86, 2024.

Paper B Suvanto M., Wahde M., and Della Vedova M. L., *Part-of-speech Taggers Make Errors on Unambiguous Sentences*. Lecture Notes in Artificial Intelligence (LNAI), in press (2025).

Paper C Wahde M., Della Vedova M. L., Virgolin M., and Suvanto M, *An interpretable method for automated classification of spoken transcripts and written text*. Evolutionary Intelligence, vol. 17, pp. 609-621, 2024.

Paper D Suvanto M. and Wahde M., *Improving glass-box sentiment classification via feature set extension*. Manuscript submitted (2025).

Technical terms

Table of contents

Chapter 1

Introduction and motivation

This thesis explores glass-box methods in **natural language processing (NLP)**. Two specific areas in NLP are considered, namely **text classification** and **part-of-speech (POS) tagging** (i.e., the problem of assigning word class labels to words in a text). The work described herein emphasizes the fundamental aspects of the two areas considered, rather than focusing on specific applications.

A subfield of **artificial intelligence (AI)**, NLP is focused on computational processing of human language, in the form of text or speech. Methods from this field can be used to detect patterns and extract information from text documents, and to automate processes traditionally requiring a considerable amount of human labor. The presence of language in most aspects of human activity makes NLP relevant across numerous domains. Some examples of these applications include meaning extraction from text (e.g., sentiment analysis; see Chapter 5), linguistic analysis (e.g., POS tagging; see Chapter 4), and consumer-oriented applications, (e.g., machine translation [52], chatbots [15], and writing assistants [20]).

Over the last decade, the field has advanced considerably partly because of an increased amount of available data, but more notably due to developments in **deep neural networks (DNNs)** and their use in NLP [28], in particular in the form of **large language models (LLMs)**. These models consist of huge neural networks pretrained on large amounts of data, a process which provides them with a foundation of language structure and syntax. They are fine-tuned for various downstream tasks, that is, specific tasks where this broad base of language knowledge can be utilized. A central task involves text generation, which is used in, for example, chatbot applications (e.g., ChatGPT), text summarization, and writing assistants. Additionally, DNN-based systems are applied in natural language understanding, for example,

in question-answering, information extraction, and text classification. For a detailed overview of the architectures and applications of LLMs see [14, 53].

The improvements in DNNs have transformed the field of AI and new applications using the most recent technology are constantly emerging. However, DNN-based methods also have several drawbacks, some of which are also relevant in NLP, such as reinforcement of biases from data [39] and their tendency to sometimes produce factually incorrect outputs, often referred to as hallucinations[1] in LLMs [42].

Another major drawback, and the most important one from the viewpoint of this thesis, is that DNNs are **black-box models** whose inner mechanisms are opaque to the human observer due to their highly distributed and non-linear computations, and their typically large size, some models having trillions of parameters [11, 51]. In many scenarios, the lack of transparency of these models is not a problem. However, there are many domains that deal with high-stakes decision making where a clear explanation of the outcome is crucial for trusting the model [29, 45]. Some examples include the medical domain, where NLP can be employed for supporting clinical decision-making [17], and in the legal domain, where NLP may be a useful tool for several tasks [27].

In high-stakes scenarios, **glass-box models** provide an alternative to black-box models. Glass-box models are interpretable due to their transparent mechanisms, and their internal workings can therefore be inspected in order to obtain a reliable explanation for a given output. The ability to provide a faithful explanation of a model's output is referred to as model **interpretability**. In our viewpoint, also expressed by [45], interpretability is distinct from model **explainability**. We work with the definition that explainability refers to explanations of black-box models, which are typically formulated from analysis of the model behavior post-hoc; see Chapter 2, where the difference between these two concepts and their relation to black-box and glass-box models is further described.

The work in this thesis is strongly motivated by the need for interpretability in NLP methods applied in high-stakes decision-making scenarios. Approaches to improve a glass-box method in text classification are explored and evaluated over two text classification tasks: Classifying text as spoken or written, and classifying text sentiment. The task of POS tagging is evaluated over novel data sets, and an interpretable rule-based approach is considered for correcting erroneous tagging results. The topics are studied from a general standpoint, paving the way for applications in high-stakes scenarios.

[1]The term *hallucination* is somewhat of a misnomer [33], but it will be used here as it is the de facto accepted terminology.

1.1 Research questions

The main research questions addressed in this thesis are as follows:

RQ1 *Does the reported performance of POS taggers reflect the true performance on previously unseen data?*

RQ2 *Can POS tagging errors be corrected with a rule-based approach?*

RQ3 *Can linear text classification be improved such that its performance is comparable to the performance of state-of-the-art DNNs?*

RQ4 *Can feature weights learned from external data sets improve linear text classification?*

RQ1 questions a common notion of POS tagging being a solved problem [22, 34] and RQ2 considers an approach for correcting errors discovered in POS tagging. RQ3 and RQ4 are focused on various improvements that can be made with respect to linear text classifiers.

1.2 Outline and author's contributions

Chapter 2 outlines the differences between glass-box and black-box models, providing more insight into the lack of transparency in the latter and the importance of interpretability in high-stakes applications. Chapter 3 describes the various data sets used for Papers A–D and (n-gram) featurization of text into a format that is readable by a machine. Chapter 4 focuses on the POS tagging task and the work carried out in Papers A and B. The text classification tasks in Papers C and D are described in Chapter 5. A conclusion to the thesis is given in Chapter 6, followed by a summary of the included Papers A–D in Chapter 7.

The author contributed to the research articles as follows: As main author in Papers B and D, and as co-author in Papers A and C. The author also carried out most of the data processing and analysis in Papers A, B and D. The author's contribution to the analysis in Paper C was more limited.

Chapter 2

Model interpretability

The pace of development and deployment of AI applications has increased considerably in the past decade. These applications emerge in a wide variety of fields, both in the private sector [19] and the public sector [6]. Interactions with AI systems are increasingly common as they are applied in many tools used daily, for example, in search engines, in online marketing, or in customer service. Essentially, AI methods can be applied in any process that produces some form of data. Here, we focus specifically on examples involving NLP applied to text data sets.

As mentioned in Chapter 1, one of the reasons for the advancements in AI is the use of neural models based on DNNs, for example, pretrained LLMs in the context of NLP. The performance exhibited by such models has transformed the AI field, creating new applications such as, for example, coding assistants, translation services, and so on. Another reason for the extensive use of DNNs, in particular LLMs, is the increased availability of such models. The high performance of these models is to a great extent due to the pretraining phase, where a model is trained on a selected task (e.g., predicting the next token in a sequence) using a very large data set. This process, including the data collection, is often time-consuming and resource intensive. Pretrained models can then be fine-tuned for use in a specific task, eliminating the need to repeat the computationally expensive training procedure.

However, as neural models are essentially black-boxes, their overall good performance comes with a major disadvantage, namely their almost entire lack of interpretability. Model interpretability is an important factor in retrieving reliable explanations of a decision-making process, which can make systems more trustworthy and increase their accountability [29, 45]. Additionally, interpretability can be helpful in detecting and correcting system

errors [45].

The lack of interpretability is especially relevant in so-called high-stakes decision-making, where even small errors may have severe consequences. The use of black-box AI in fields that deal with high-stakes decision making, such as in healthcare or legal applications, is problematic and has raised many discussions [29, 45]. On the one hand, in such fields, AI tools can be a valuable asset and save time, especially in scenarios where there is a demand that is difficult to meet, for example, due to a lack of personnel. For example, a chatbot service offered by healthcare providers can give initial guidance to patients [2]. Similarly, a summarization tool can generate an overview of legal documents [3]. On the other hand, uncontrolled and incorrect outputs can have unexpected repercussions in such situations [36] and a lack of explanation for the output can weaken the trust in the systems. Therefore, explainability and interpretability have become areas of focus in AI [29, 45].

This thesis is focused on human-interpretable (glass-box) methods which have a transparent structure and thus are capable of offering a reliable explanation of their decision-making. Classical methods in AI are often interpretable (albeit to a varying degree), even though they were not explicitly designed for that purpose, as interpretability was arguably less relevant prior to the appearance of DNN-based models.

By contrast, explaining the decision-making process of black-box models, such as DNNs, is typically done post-hoc[1] from the model *output* rather than from the model *structure*. Some commonly employed techniques used for obtaining explanations from black-boxes are, for example, LIME [44], SHAP [12], and saliency [48]. However, such methods do not always provide meaningful explanations and they may be misleading or even incorrect [49]. Moreover, different methods used for explaining the same model often give conflicting results [26].

It should also be noted, however, that there are many cases where model interpretability is not needed. For example, as noted in [38], applications that have been well studied typically may not gain any additional value from interpretability, such as in optical character recognition or in speech recognition. Indeed, the specific tasks in this thesis (e.g., detecting movie review sentiments) do not require interpretability, and the aim of this thesis is instead to *lay a foundation* for building applications where interpretability *is* relevant.

[1] An emerging approach for explaining DNNs is *mechanistic interpretability*, an attempt to decompose (or *reverse engineer*) DNNs into a human-understandable form. At the time of writing this thesis, this approach is still being developed [50] and it is not clear to what degree it will impact the field.

Chapter 3

Data and features

This chapter introduces the data sets used in Papers A–D in Section 3.1, the procedures for preprocessing in Section 3.2, and feature extraction in Section 3.3.

Every day, massive amounts of natural language data are documented. The data consist of formally written text, such as articles in news outlets or online encyclopedias, as well as less formal text produced through online communication, such as e-mails or forum discussions. Language data can also be recorded and transcribed from, for example, radio shows and phone calls. Much of it is in the public domain, so that it can be freely accessed and therefore also utilized for text processing. The increased availability of data[1] has been essential in improving the performance of the methods used in NLP. Many methods in NLP use statistical approaches, which are data-driven, meaning that they rely heavily on large quantities of data.

In data-driven methods, a portion of the data is typically used for validating model performance. A common approach for this is **holdout validation** wherein data are generally arranged into three sets: (1) a training set used to train models, (2) a validation set to select the best-performing model, and (3) a test set for evaluating the performance of the selected model. Another approach is **k-fold cross-validation**, where the data are divided into k subsets. Then, $k-1$ subsets are used in training and the k^{th} set for validation in turns throughout the training phase. In this thesis, holdout validation was used to train and evaluate the models, and therefore the majority of the data sets were split into to the aforementioned three portions.

The data sets need to be converted into a format that is suitable for computational processing, a procedure known as text preprocessing. This pro-

[1] However, most of the data pertain to languages with many speakers; small languages are less represented in the available data [4].

cedure generally involves several steps, such as tokenization, spell checking, spelling normalization, etc., further described in Section 3.2. Preprocessing is followed by feature extraction, where the text is converted to either abstract encodings (e.g., word embeddings in DNN-based models) or explicit representations (e.g., n-grams used in the interpretable models considered in this thesis).

3.1 Data sets

The data sets used in the POS tagging and text classification tasks in Papers A–D are briefly described in this section. Some of the data sets are well established with previously reported performance metrics while others were collected for the specific tasks considered in this thesis. Papers C and D deal with binary text classification, meaning that there are two classes, here denoted as Class 0 and Class 1. However, the categories represented by these classes differ among the two papers and are specified where needed.

3.1.1 Standard POS data sets

Data sets for POS tagging, including the ones described here, typically consist of tokenized text (see Section 3.2), where each token has been annotated with a POS tag. Annotations are done either manually or via an automated process, and various tag sets offering different levels of granularity have been used in different data sets. Two data sets that are commonly used when training taggers, and thus function as relevant benchmarks in Papers A and B, are the Wall Street Journal (WSJ) section from the Penn Treebank (PTB) data set [35] and the Brown Corpus data set [21]. The PTB set consists of 2,499 WSJ stories and has around 1 million annotated tokens [35]. The Brown Corpus has 500 text samples and a total of approximately 1 million annotated tokens [21].

Both of these data sets have been annotated manually, using different tag sets to label the tokens; PTB uses a tag set of 45 tags and Brown uses 87 tags. However, we use a mapping to a smaller tag set (the so-called Universal tag set [43]), which enables the comparison of tagger performance over the two different data sets. The text in the data sets was written over 30 years ago (PTB) and over 50 years ago (Brown), suggesting that their vocabulary may be partially outdated, a problem which is discussed in Papers A and B.

Data set	Sentences	Tagged tokens
Data set in Paper A	2,227	2,227
Data set in Paper B	1,123	5,850

Table 3.1: Some basic data regarding the two newly collected data sets for POS tagging.

	Size (number of items)			
Data set	Training	Validation	Test	Total
Spoken vs written (Paper C)	9,743,188	1,948,639	1,948,631	13,640,458
IMDB ([31], used in Paper D)	20,000	5,000	25,000	50,000
Rotten Tomatoes (Paper D)	20,260,968	5,059,507	N/A	25,320,475

Table 3.2: The size of the subsets in each text classification data set.

3.1.2 New POS data sets

Papers A and B introduce two new POS tagging data sets that are designed to be challenging for existing taggers trained on the standard data sets described above. First, in Paper A, sample sentences for words were extracted from a web-based dictionary (Wiktionary). The resulting data set consists of 2,227 sentences, where a single word in each sentence is annotated, making it suitable for testing existing taggers. Building on this work, Paper B introduces a data set generated by tagging and selecting sentences (of spoken and written origin collected in Paper C, described below) where taggers disagree on one token instance. This second procedure results in a set of 1,123 fully tagged short sentences (on average 5.2 tokens per sentence), thus making it also suitable for training. The data set sizes are summarized in Table 3.1. Both data sets were first annotated automatically, using the Universal tag set [43], followed by manual verification of the labels and a removal of sentences with any erroneous annotations. The manual processing is time-consuming, thus limiting the final size of the data sets. In their current form these data sets are suitable for proof-of-concept analyses.

3.1.3 Spoken and written text

A data set for distinguishing between written text and spoken transcripts was generated in connection with Paper C. It consists of single sentences labeled as either spoken (Class 0) or written (Class 1). The spoken sentences were collected from transcripts from radio shows, and the written sentences from Wikipedia pages. In many cases, very short sentences lack the context to reliably be assigned either class label. Therefore, sentences of less than 5 tokens were excluded, as they could be considered to belong to either class. In total, 13,640,458 sentences were collected (of which 47% are in Class 0 and

53% in Class 1). The sentences were then split into a training set, a validation set, and a test set; see Table 3.2. The average sentence length in Class 0 is 14.79 tokens, whereas the sentences in Class 1 had an average length of 20.69 tokens, a difference which was identified as helpful in classifying the text in Paper C.

3.1.4 Movie reviews

In Paper D, text sentiment is classified using the IMDB movie review data set [31], which has been used frequently in sentiment analysis; see Paper D for results over various methods. The reviews were automatically labeled (in [31]) using a score associated with the original review. The scores were in the range 1-10 and only strongly polarized reviews were included in the data set, where a negative label (Class 0) was assigned for reviews with a score ≤ 4 and a positive label (Class 1) for score ≥ 7. An automated process for labeling data introduces a possibility of label errors [16] and, in fact, around 2.9% of the ground-truth labels in the IMDB test set are incorrect [40], possibly due to a discrepancy between the score associated with the review and the sentiment conveyed in the review text. Despite the possibility of introducing errors, this practice of labeling is common for polarity data sets extracted from reviews, due to the often large sizes of these data sets. This discrepancy has generally not been considered in studies involving the IMDB data set. Thus, in order to reliably compare our results to those found in the literature, the original labels from [31] are used in Paper D.

In [31], the data were divided into a training set (25,000 reviews) and a test set (25,000 reviews) with an even split between classes. The method in Paper D also uses a validation set, for which 20% of the training set (a fraction commonly used for validation) was extracted, resulting in a split of 20,000 (training), 5,000 (validation) and 25,000 (test) reviews, again with an even split between classes; see also Table 3.2.

The experiments carried out in Paper D also require an extended data set, which was generated by collecting and processing a large set of movie reviews from the Rotten Tomatoes platform, consisting of two sets of reviews written by critics and users, respectively (see Paper D for the exact source of the data sets). The reviews were compared to each other within these two sets and to reviews in [31], and all duplicate reviews were removed. In the Rotten Tomatoes service, critic reviews are assigned a status *rotten* or *fresh*, which were mapped to negative (Class 0) and positive (Class 1) reviews, respectively. The status and the mapped label were included in the downloaded data set. The user reviews instead included a star rating in the range $0.5-5.0$. Automated labeling was applied (as in [31]); negative reviews

(Class0) were selected from reviews with score ≤ 1.5 and positive reviews (Class1) from reviews with score ≥ 4.5. The critic and user reviews were combined into one set, resulting in a total of 25,320,475 reviews, with 21% belonging in Class 0 and 79% in Class 1 (the class imbalance was considered in the training phase, described in Chapter 5). As the extended data set was only considered for a training phase, no test set was extracted, and the data were split into a training set and a validation set; see Table 3.2.

3.2 Text preprocessing

Text preprocessing is a procedure where data are converted into a format that is suitable for computational processing. A common approach for generating data sets is to collect text from the internet via an automated process. Such text often contains parts that are irrelevant to the task at hand, e.g., markup tags, spelling errors, grammatical errors, or other inconsistencies, such as a mix of regional spelling variations. In addition to the necessity of formatting the data, an appropriate preprocessing sequence can also improve the data quality, which in turn can increase the accuracy of a model in some tasks, such as in text classification [23].

The exact preprocessing steps required varies between different data sets and different tasks, but many well established practices exist [13]. Some common preprocessing steps include removing stop words (common words that hold limited or no semantic meaning, e.g., *is*, *the*), word **stemming** or **lemmatization** (two processes that reduce words to their base form, e.g., the lemma of *dancing* is *dance*), **spelling normalization** (e.g., converting words to one spelling variety of English, or expanding common abbreviations), spelling correction, and **case folding** (e.g., converting the text to lowercase).

Another essential preprocessing step is text **tokenization**. This is the process of splitting a text into smaller units called **tokens**. When processing a training data set, it is common to form a vocabulary, i.e., the set of distinct tokens that appear in the data set. A straightforward approach is to split the text into individual words based on whitespace characters. Some tasks require text to be split into smaller subword units, which can be achieved with, for example, byte pair encoding (BPE) [47] or WordPiece [46]. This approach makes it possible to construct a smaller vocabulary that is more flexible in handling out-of-vocabulary words. On the other hand, subword tokenization may result in some information loss, for example, if words are split incorrectly [8].

In some cases, tokenization into units consisting of more than one word

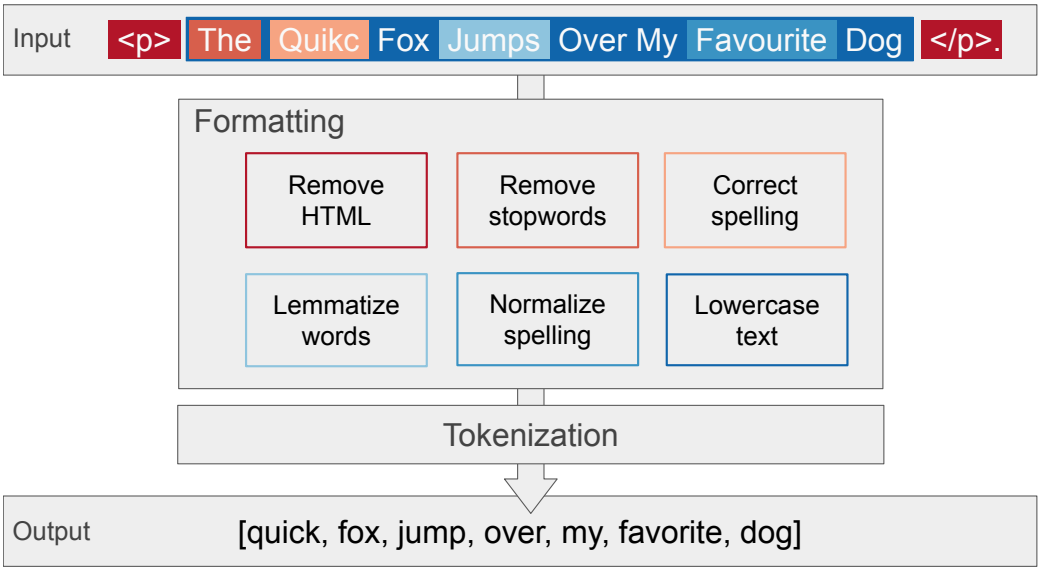

Figure 3.1: An example of a text preprocessing procedure. The different colors of the highlighted text correspond to the actions affecting them. Spelling normalization (medium blue) converts the text into American English in this case. The input to the preprocessor is a text string, and the output is a list of token strings. The output can alternatively be converted into token indices in a specified vocabulary.

can be used in order to retain more semantic information. For example, a specific meaning can be conveyed by a group of words, such as with phrasal verbs (e.g., *hang out*), compound nouns (e.g., *high school*), or idiomatic phrases (e.g., *raining cats and dogs*), where word-level splitting would result in information loss. However, maintaining larger units can also result in a more narrow and specific vocabulary that does not generalize well to other data sets. As with the previously introduced preprocessing steps, the appropriate approach for tokenization is task-dependent. A split into smaller units is common in text generation whereas in other tasks, such as POS tagging (Papers A and B), maintaining word-level information is preferred.

In the text classification tasks (Papers C and D), the preprocessing steps were (1) removing markup tags, (2) lowercasing, (3) removing consecutive whitespaces, and (4) tokenization into words, resolving punctuation marks as separate tokens. These preprocessing steps result in a more precise vocabulary. For example, by applying lowercasing, more instances of words can be captured, e.g., in the sentence *this movie was GREAT*, the word *GREAT* is mapped to the same token as *great*, a more common spelling variation in the data set. This preprocessing approach is suitable for the feature extraction applied in this case, described below.

3.3 Text features

Text features are representations that are used as the model input in the various NLP tasks. They typically aim to capture some meaningful aspects of text, such as sentiment polarity or semantic similarity to other texts. In Papers C and D, the text was represented as n-**gram** features. Other methods for feature representations exist as well, most notably **word embeddings**, such as word2vec, GloVe, ELMo, and many more (see [1] for details), which are multidimensional vector representations of text. Word embeddings are exposed to words used in a variety of contexts during their training phase, allowing them to capture contextual and linguistic information. While they are an integral part in encoding the input of many state-of-the-art neural NLP models, word embeddings lack the type of explicit interpretability that is the primary focus explored in this thesis. In addition to the n-gram features used in Papers C and D, Paper B makes use of some morphosyntactic features commonly used in POS tagging.

3.3.1 n-grams

Text representations in the form of n-gram features were used frequently in NLP applications prior to the advent of neural models and they are still relevant in many NLP tasks [30]. These features are constructed from sequences of adjacent tokens[2]. The basic case of $n = 1$ involves individual tokens (also called *unigrams*), and expanding this representation to $n = 2$ yields token pairs (also called *bigrams*). An illustration of different values of n is shown in Fig. 3.2. For a given value of n, a set of $T + 1 - n$ features can be extracted from a text with T tokens. The feature representation for a text can be combined from each of the n-gram sets, up to a given maximum value of n.

As can be seen in Fig. 3.2, the value of n can in principle be extended up to T, where the case of $n = T$ is the entire text. The feature representation of a text will include a total of $T(T + 1)/2$ features when using all the n-gram sets for each n in the range 1 to T. However, this approach is not feasible in practice as the representation of the text becomes too large to handle computationally, since one must often process several million tokens (see Table 3.2). For example, a data set of 1 million tokens would result in text representations of more than 500 billion features. The processing of these features is restricted by the computational memory available.

[2]We construct n-grams from tokenized *words*, but note that the term may also refer to n-grams consisting of *characters* within a word.

$n = 1$: (the), (fox), (jumped), (quickly), (!)

$n = 2$: (the fox), (fox jumped), (jumped quickly), (quickly !)

$n = 3$: (the fox jumped), (fox jumped quickly), (jumped quickly !)

$n = 4$: (the fox jumped quickly), (fox jumped quickly !)

$n = 5$: (the fox jumped quickly !)

*Figure 3.2: Examples of n-gram features extracted from the sentence "*The fox jumped quickly!*" that was first lowercased and tokenized. For the case $n = 5$ the resulting n-gram is the entire sentence.*

Furthermore, larger n-grams will appear less often in out-of-sample texts. Thus, the training procedure is likely to overfit such features, leading to a drop in performance, for example, when applying a text classification method on previously unseen data. A common practice is therefore to use a relatively small value of n and to include only n-grams that appear with sufficient frequency in the training set. Both of these practices have been implemented in Papers C and D.

A vocabulary of n-grams can be constructed from the training set such that each feature is associated with a unique integer index. A text featurized in that way will thus consist of a vector of feature indices, where each value represents the count (i.e., the number of instances) of the feature corresponding to the index. These vectors can be normalized and smoothing techniques can be applied to handle out-of-vocabulary instances. However, in the text classification task in Paper D, the features were instead read directly from a dictionary (in the form of a hash table, with $O(1)$ lookup time), which is efficient for the large set of n-grams used in the task.

The aforementioned inability to realistically extend to a large enough n is a major drawback that limits the possibilities of capturing long-range context and the relationships between words in the text. Despite these limitations, n-grams are suitable for many applications. As shown in Papers C and D, using a moderate size of n can result in sufficient representations and well-performing models. Furthermore, n-grams are arguably easy to interpret, as is illustrated in Papers C and D.

3.3.2 Other text features

Besides the previously mentioned categories of text representations (n-grams and word embeddings) various other features can be extracted from text. Such features are, for example, word length, POS tags, dependency relations, or synonymous words. In POS taggers, for example, it is common to use

previously predicted tags, surrounding words, and affixes (e.g., prefixes and suffixes) as additional features. In the corrective tagging approach introduced in Paper B, surrounding words, predicted tags, and suffixes were used as features for the rule-based approach.

Chapter 4

Part-of-speech tagging

The task of POS tagging is introduced in this chapter. The state-of-the-art taggers described in Section 4.1 were used as a baseline for tagger performance in Papers A and B. An overview and discussion of the results in Papers A and B is given in Section 4.2, where the research questions RQ1 and RQ2 are also addressed.

4.1 Methods

Automated POS tagging is a **sequence labeling** task where each word in a sequence of words (such as a sentence) is assigned a POS tag, i.e., a label that determines a word class and, in some fine-grained tag sets, more grammatical details (e.g., verb forms or noun cases). In many cases, a word can belong to multiple word classes, meaning that the correct tag for a word must be determined from the context in which it appears. An example of a tagged sentence is shown in Fig. 4.1. In this example, the word *brown* functions as an adjective. In different sentences, it can also be a verb (e.g., *Can you brown the butter?*), a noun (e.g., *I painted the walls a warm brown*), or, when capitalized, a proper noun (e.g., *The tagger was trained on the Brown corpus*).

Tagging text with POS tags is a common intermediate step in various applications. The resulting grammatical information can be useful in, for example, word-sense disambiguation, machine translation, and keyword extraction. While the popularity of POS tagging has, to some degree, declined due to the rise of end-to-end neural networks, it can still be relevant in neural applications as well [24, 54]. In POS tagging, the focus is on tagging individual words, but there are other sequence labeling tasks where larger portions of

Sentence:	The	brown	fox	jumps	over	the	chicken	coop
Tags:	DET	ADJ	NOUN	VERB	ADP	DET	NOUN	NOUN

Figure 4.1: An example of a POS-tagged sentence. Each word has been assigned a tag from the Universal tag set [43]. In this sentence, the last two words form a compound noun chicken coop*, for which a common convention is to tag both parts as a* NOUN.

text are tagged. These include **named-entity recognition** (NER), where specific actors or objects referred to by a name are labeled (often involving proper nouns, e.g., the previous example of *Brown*), or text **chunking**, where words are grouped into meaningful units, such as noun phrases (e.g., *the brown fox* and *the chicken coop* in Fig. 4.1). These two tasks also typically use POS tagging as an intermediate step.

4.1.1 Baseline POS taggers

A simple baseline for POS tagging is the so-called **unigram tagger**, where words are tagged with their most frequent tag in the training corpus. This method typically reaches a test accuracy of about $0.85 - 0.93$, depending on which data set and tag set is used. This rather high baseline accuracy is possible to obtain since many words are either associated with just one tag, or are frequently used as only one of their possible word classes. For example, in the Brown corpus, the (lowercased) word *brown* appears 65 times in total, of which 61 instances are adjectives and 4 instances are nouns. Therefore, simply tagging the word as an adjective will be correct in most cases. For many words, however, the difference in tag frequency is not so distinct. Such cases can be resolved using a more advanced POS tagging method that considers the context of the word.

Standard methods for POS tagging can be split into two broad categories: (1) manually defined taggers, such as the ENGTWOL/CG tagger, which consists of linguistic rules, and (2) automatically trained taggers, which can be rule-based, such as the Brill tagger, statistical, such as the Perceptron, Stanford, and Hunpos taggers, or DNN-based; see Papers A and B for references. Papers A and B use rule-based and statistical taggers (herein referred to as standard taggers) as a baseline. Notably, the performance difference between the methods, including DNNs, is not very large. Taggers trained with these methods typically perform very well, with reported test accuracies being around 0.97. These results are typically obtained by running a tagger over a held-out test set from one of the standard POS data sets described in Section 3.1. With this high performance, the task of POS tagging is often considered a *solved* problem. However, as will be discussed in Section 4.2

(in regards to RQ1) and in Papers A and B, such a categorical statement is somewhat premature.

4.1.2 Ambiguities in POS tagging

In many cases, there is only one correct POS tag sequence for a given sentence (and given tag set). However, there are also some inherently ambiguous cases where more than one tag sequence can be correct. For example, tagging open compound nouns, which consist of more than one word separated by whitespace characters, may not be always be straightforward. A common convention to tag such nouns is to assign the tag *NOUN* to each part of the compound noun, as can be seen in the word *chicken coop* in Fig. 4.1. This convention also applies to compound nouns where the first word is typically an adjective. For example, the noun *hot dog* would still be assigned the sequence *NOUN NOUN*. In the POS-tagged sentence in Fig. 4.1, it is clear that *brown fox* is not a compound noun, but if the adjective were to be changed to *red*, the example becomes ambiguous, as *red fox* is, in fact, a compound noun (i.e., the English term for a species of animal). The sequences *ADJ NOUN* and *NOUN NOUN* are equally correct options for *red fox*. The problem of tagging compound nouns and some other ambiguities[1] related to POS tagging arising from tagging conventions, tokenization, and the past and present participle forms of verbs are further discussed in Paper B. These examples illustrate a core problem with POS tagging as a task, namely that assigning one POS tag to each word cannot always be done unambiguously.

4.1.3 Corrective rule-based approach

In Papers A and B, we show that when standard POS taggers are applied to our new data sets, devised so as to be challenging, their performance drops considerably from the reported test accuracies, as also discussed in Section 4.2. Here, it should be noted that the sentences evaluated in Paper B were, in fact, selected such that they contain none of the ambiguities mentioned above, meaning that the standard taggers make errors even in straightforward and unambiguous cases. These errors can often be corrected with some grammatical knowledge. Thus, a corrective rule set was generated in Paper B to correct the tagger outputs.

[1]These examples are drawn from the English language, and are by no means universal across different languages. For example, in Swedish and Finnish, compound nouns are generally closed. Likewise, different languages may also have ambiguities that do not apply to English.

Sentence:	BOS	she	started	to	**laugh**	.	EOS
Tags:	BOS	PRON	VERB	PRT	**NOUN**	.	EOS

```
IF token(-1) = to
IF tokenSuffix(-2) = -ed
IF POS(-3) = PRON
THEN POS(0) = VERB
```

*Figure 4.2: An example of a corrective rule generated from the sentence "*she started to laugh*". The tokens BOS and EOS are appended to each training sentence to mark the beginning and end of the sentence. In this example, the targeted token with position 0 is* laugh*, which was incorrectly tagged as a* NOUN*. The generated rule assigns the correct tag* VERB*. Note that this rule is generated in the first phase but it is subject to potential removal in the second phase where the rule set is refined.*

The rule-based approach for correcting the output of standard taggers was proposed in Paper B, where a more detailed description of the method can be found. A rule set was automatically generated in two steps: First, rules were exhaustively generated from sentences in a training set and, secondly, the exhaustive set of rules was reduced to only the most generic ones, which are more likely to be applicable also to other data sets. The resulting set of rules can be applied to the output of any existing tagger, if its output tags are taken from the same tag set as in the training set.

The rules are structured as `if-then` rules that reassign the tag of the incorrectly tagged word, shown in Fig. 4.2. As mentioned above, as the first step, a set of candidate rules is generated from a training set that should consist of sentences where each token has been assigned a tag, and where, in our case, precisely one word has been incorrectly tagged, but its ground truth tag is known. The rule parameters are token spellings, assigned POS tags, and token suffixes, which are extracted from tokens surrounding the targeted token, marked with the index 0. Preceding tokens are assigned negative indices whereas succeeding tokens are assigned positive indices, as shown in Fig. 4.2. As also seen in that figure, the sentence contains the additional tokens BOS, which marks the beginning of a sentence, and EOS, which marks the end of a sentence. These markers are appended to the sentences and are considered as surrounding tokens when extracting parameters for the candidate rules.

After the set of rules has been exhaustively generated, it will contain many rules that are very specific to the training sentences, and rules that can introduce new errors. Therefore, the set is evaluated on an external data set, and any rules that trigger less than m times or produce at least k errors are removed. The final set of rules can then be used to correct the output of a

Tagger	Accuracy
Brill	0.473
Hunpos	0.482
Stanford	0.673
Perceptron	0.561
DNN-based	0.868

Table 4.1: The accuracy over the new test set from Paper A for the selected taggers. The accuracy is measured by considering only the tokens that have a ground truth label (in this case, one per sentence).

	Training set (Paper B)		Test set (reduced) (Paper A)	
Tagger	Initial accuracy	New accuracy	Initial accuracy	New accuracy
Brill	0.5156	0.8272	0.4647	0.5037
Hunpos	0.5209	0.8540	0.4963	0.5287
Stanford	0.8362	0.9341	0.6858	0.7032
Perceptron	0.4809	0.7845	0.5736	0.5894
DNN-based	0.9457	0.9697	0.8845	0.8878

Table 4.2: The accuracy of the selected classifiers over the new training set and test set before and after applying the generated corrective rule set to the tagger output. The accuracy is measured by considering a single target token in each sentence, that is, the token that was incorrectly tagged by at least one of the taggers; see also Paper B. Note that the test set from Paper A has been reduced here so as to contain only unambiguous sentences.

tagger by evaluating each token in a sentence against the rules, and applying the corrective action to tokens when all conditions in a rule are fulfilled. The rules are evaluated in the order that they appear in the set, and once a rule is triggered for a token, the process is terminated and no further rules are evaluated for the specific token.

4.2 Observations

An overview of the tagger performance over the two new data sets in Papers A and B is shown in Tables 4.1 and 4.2. It is important to note that the four standard taggers (Brill, Hunpos, Stanford, and Perceptron) were used in the data selection process that explicitly aimed at finding challenging examples. Thus, the large decrease from their expected performance of around 0.97 is not so surprising. However, when inspecting the performance of the DNN-based tagger, which was *not* involved in the data selection process, we can observe a true out-of-sample test. In this case, the decrease from the reported tagger accuracy of around 0.97 is substantial: For the first data set (see Table 4.1), its accuracy is around 0.1 below its reported accuracy. The performance (before applying corrective rules) over the second set is quite a

bit better (0.95; see Table 4.2), but still somewhat below the reported accuracy. Thus, in regards to RQ1, these results show that the true performance of POS taggers does indeed decrease when they are applied over a challenging data set.

Paper A discusses reasons for the tagger failures, which include, for example, effects of temporal drift (i.e., the fact that the meaning of words changes over time), a lack of general grammatical knowledge (such as the fact that a complete sentence should normally contain a verb), and similarities in word morphology or surrounding context. An example of the last type can be observed in the tagged output for the two sentences (from the data set in Paper A), (i) *He was hired to bus tables [...]*, and (ii) *I go to church every day*. In sentence (i), three of the POS taggers incorrectly tagged the word *bus* as a *NOUN*, when it should be a *VERB*, and in sentence (ii), the word *church* was incorrectly tagged as a *VERB* by two of the POS taggers, when it should be a *NOUN*. In both cases, the incorrectly tagged word is preceded by the particle *to*. Further examples that were challenging for the taggers are shown in Table 4.3.

As mentioned above in Section 4.1, there are many systems that rely on automated POS tagging. In those cases, an incorrectly POS tagged sentence can propagate additional errors downstream. The corrective approach from Paper B focused specifically on unambiguous cases and the results, summarized in Table 4.2, show that such an approach can indeed be used to correct the output of the taggers, thus providing a preliminary positive answer to RQ2. As the approach was evaluated with a small data set where the sentences were short, these results should be seen as a proof-of-concept. Many of the examples in the test set (Paper A), such as the two examples mentioned above with the *to* particle, were not associated with a rule that could correctly reassign the POS tags. In these cases, suitable rules were either not generated at all (as the training set may not have contained relevant examples for those cases), or were filtered out in the reduction phase (see the last paragraph in Section 4.1.3). The final rule set also included some rules that generated further errors, though the net result was positive overall (see Paper B). An obvious way to improve the rules and to make them applicable in more sentences is to use a larger training set and a fine-grained tag set, which can provide more valuable grammatical information (e.g., cases or tenses).

Sentence	Actual	Brill	Hunpos	Stanford	Perceptron	DNN
Data set in Paper A						
The goat climbed up the **craggy** rocks .	ADJ	**NOUN**	ADJ	**NOUN**	**NOUN**	ADJ
Pass the wine , would you , **darling** ?	NOUN	NOUN	NOUN	NOUN	**VERB**	**VERB**
These cucumbers **pickle** very well .	VERB	**NOUN**	VERB	**NOUN**	VERB	VERB
When they retired , they hoped to **winter** in Florida .	VERB	VERB	**NOUN**	**NOUN**	VERB	**NOUN**
Snow **slides** down the side of a mountain .	VERB	**NOUN**	VERB	**NOUN**	VERB	VERB
Data set in Paper B						
it was a **musical** .	NOUN	NOUN	**ADJ**	NOUN	**ADJ**	NOUN
however , we are **equals** .	NOUN	**VERB**	**VERB**	NOUN	NOUN	NOUN
it does **sound** incredible .	VERB	**NOUN**	**ADJ**	VERB	**ADJ**	VERB
then the telephone **rang** .	VERB	VERB	VERB	VERB	**NOUN**	**NOUN**
ratings **matter** on television .	VERB	**NOUN**	VERB	VERB	**NOUN**	VERB

Table 4.3: Some example sentences from the two challenging POS data sets. The token under consideration is shown underlined and in bold. The ground truth tag is shown in the second column, and the tags assigned by the different taggers are given in the remaining columns.

Chapter 5

Text classification

The task of text classification, explored in Papers C and D, is described in this chapter. Various methods from the literature as well as the linear text classifier used in Papers C and D are described briefly in Section 5.1, including the adjustments made to the classifier that improve the classification accuracy. Section 5.2 includes a discussion on the results, where the research questions RQ3 and RQ4 are considered.

5.1 Methods

Text classification is the task of automatically categorizing text. It can be used to sort large quantities of text into predefined classes, or to gain some general insight of the text content. Some applications where text categories (or classes) are commonly considered are sentiment analysis (described in more detail below), language identification, spam-filters, textual entailment (i.e., a task where relations between texts are detected), fake news detection, and many more, as described in [25, 37]. In binary classification there are only two class labels (e.g., Class 0 and Class 1), such as in the two tasks considered in this thesis, whereas multi-label classification tasks involve more than two classes, such as in, for example, fine-grained emotion detection.

The classification task in Paper C focused on categorizing text as spoken (i.e., transcribed text) or written. Such categorization can be useful in, for example, determining the suitability of a sentence for a formally written text, such as an academic report. An example of a sentence that is initially classified as spoken, and then reformulated to match the written class, is shown in Fig. 5.1. The features that contribute most to the classification are highlighted using the visualization tool introduced in Paper C. Such a tool

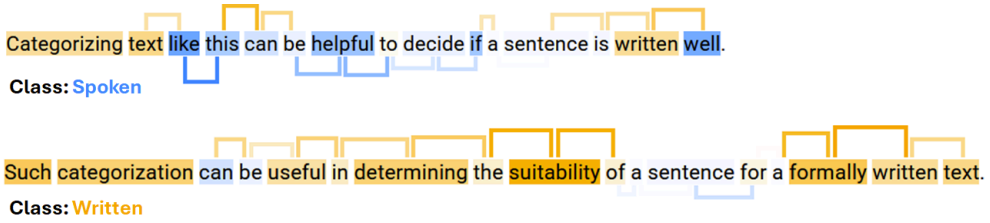

Figure 5.1: An example of the text classifier using unigram and bigram features. The first sentence is classified as spoken*, as it contains multiple features typically associated with spoken language (highlighted in blue). The second sentence is a reformulation of the first one, and is classified as* written*, since most of its features are now more associated with the written class (highlighted in yellow). The visualization was generated with the online tool:* `https://aaiserver.m2.chalmers.se/spoken_vs_written_tool`

can help in identifying the specific parts of a text that must be reformulated such that they match one of the given classes.

Text **sentiment** classification (also known as **sentiment analysis**) is a common special case, which is applicable in many domains [7], for example, in market research, medical text analysis, and financial trading. The task was explored in Paper D, where the polarity (i.e., positive or negative sentiment) of movie reviews was considered. In general, sentiment analysis can involve other NLP tasks [10], such as word-sense disambiguation, sarcasm detection, or anaphora resolution. Another variety of the task is **aspect-based sentiment analysis**, where the focus is on extracting specific aspects and identifying the associated opinions [9].

Several methods have been developed and used for generating text classifiers. In Papers C and D, a linear classifier was used. It was generated via **supervised learning**, where the algorithm uses ground truth labels from the training set to guide its learning process. Many other commonly used methods in text classification are also based on supervised learning. An alternative method is **unsupervised learning**, where the algorithm instead groups similar texts together without any labeled examples. Another approach is also semi-supervised learning, which combines the two learning methods. For example, the pretraining phase of LLMs is typically unsupervised, followed by fine-tuning (for text classification in this case), which is often done with a supervised learning phase.

5.1.1 Classical methods

Prior to the advent of DNN-based methods, text classification was typically carried out using classical methods, such as support vector machines

(SVMs), perceptrons (also discussed further in Section 5.1.3), naïve Bayes (NB) classifiers, k-nearest neighbor (kNN) classifiers, decision trees, and logistic regression; see also Papers C and D. In general, such methods have a good balance between training complexity and performance. The performance of these methods usually varies between specific tasks and data sets; for an overview, see Table 5.1.

5.1.2 Neural methods

Early neural methods used shallow networks, which were used in parallel with the classical methods introduced above. Such methods typically achieved accuracies similar to those of the classical methods. The introduction of DNNs in the recent years led to a leap in performance and thus established the current state-of-the-art in text classification. Such methods include, for example, recurrent neural networks (RNNs), in particular its variant of long short-term memory (LSTM), convolutional neural networks (CNNs), as well as pretrained LLMs, of which especially BERT [18] is often employed for classification. Compared to the classical methods, DNN-based methods typically perform better (see Table 5.1) but have a more complex training procedure. While the performance reported for LLMs specifically is typically quite high, such results may pose an unfair comparison due to data contamination: As LLMs are trained on a huge amount of data, it is possible that the selected test set has been included in the training data [32].

5.1.3 Interpretable methods

This section focuses on methods that are explicitly developed to be interpretable, and in particular, the method used in Papers C and D. It should be noted that many of the classical methods introduced above are typically transparent to some degree, and are generally more interpretable than DNN-based methods. However, this is a more of a coincidence rather than an intentional choice. Prior to the introduction of DNNs, model interpretability was rarely considered to be important. As the terms *interpretability* and *explainability* are often used interchangeably, several studies on *interpretable text classification* use methods based on DNNs (using methods such as LIME [44]). However, as previously discussed in Chapter 2, the explanations of DNN outputs do not exhibit the type of interpretability that this thesis focuses on, which is why the DNN-based methods used in such studies are instead considered to be part of the neural methods described above.

Despite the dominance of DNN-based models, there have been some studies focusing on explicit interpretability as per the definition used in this thesis.

The methods used in these studies are, for example, hidden Markov models (HMMs) [41], and a system based on propositional logic that is referred to as a **Tsetlin machine** [5].

Method used in Papers C and D

The method for text classification in Papers C and D is a linear text classifier based on n-gram features. A brief description of the general procedure is given in this section. The classification and optimization procedure for each task is described in full detail in Papers C and D. Those descriptions also include additional concepts used in the classification, such as a length adjustment term introduced in Paper C. The general procedure for classifying a text (herein also referred to as an *item*), that is, either a sentence (as in Paper C) or a longer text (as in Paper D), begins by computing a classification sum

$$s = \alpha + \sum_{i=1}^{V} w_i f_i \tag{5.1}$$

where α is a bias term and w_i are the feature weights in a vocabulary with the size V. The feature values f_i are the frequencies of the features in an item. The predicted class $\hat{C}$ of a text then is determined by the step function

$$\hat{C} = \begin{cases} 1, & s \geq 0 \\ 0 & s < 0 \end{cases} \tag{5.2}$$

When generating the classifier, the weights are initialized in a given range, here typically taken as $[-1, 1]$. A suitable performance measure (such as accuracy or F1 score) is chosen for evaluating the classifier performance. Each training item is classified (as described above) using the current feature weights, and the feature errors are computed as

$$e_i = \frac{1}{\gamma_i} \sum_{j=1}^{N} v_{ij}(\hat{C}_j - C_j), i = 1, ..., V \tag{5.3}$$

where γ_i is the total number of instances of feature i in the training set, N is the number of items in the training set, and v_{ij} is the number of instances of the feature i in the training item j. The true class of the item j is denoted by C_j and $\hat{C}_j$ is the inferred class. The weights are then updated

$$w_i \leftarrow w_i - \eta e_i \tag{5.4}$$

where η is a learning rate. The bias term is updated in a similar manner, but with a simplified error computation; see Papers C and D for details. This

weight update procedure is repeated over several epochs and the classifier with the best performance (i.e., accuracy or F1 score) over the validation set is selected as the final classifier.

This procedure for training a linear classifier resembles both gradient descent in linear regression, and the perceptron algorithm. However, there are also some differences: In linear regression, a squared weighted feature sum is used to compute the error (i.e., mean squared error), whereas in the method described above, the error is computed from the discrete class assignment, and the impact of each feature is considered individually. In the standard perceptron training algorithm, feature weights are updated after each training item is classified, a step which involves shuffling the training data at the beginning of each epoch in order to avoid bias towards a specific permutation of the data set. The method described above (and in Papers C and D) instead updates the feature weights once after the entire training set has been classified.

Extension method in Paper D

A method for improving the classification by strongly extending the size of the feature set was proposed in Paper D. The concept is similar to the idea of pretraining, but the implementation is vastly different. In tasks where the training set is relatively small, the feature set may not cover many of the possible features in out-of-sample data (i.e., data outside of the training set). In some cases, these features can be essential for predicting the class of the text. In Paper D, it was found that when classifying text with a linear classifier, text items with a larger **coverage ratio** (i.e., the number of all *covered* features divided by the number of all *potential* features in a text) were more often correctly classified. Therefore, it was hypothesizes that by expanding the size of the feature set, as described below, the out-of-sample performance of a linear classifier could be improved.

The approach uses the linear classification method described above, but combines the weights from (i) the feature set of the original training set, and (ii) the feature set of an extended training set. The feature set of the final classifier is constructed in two steps: First, weights of the features that are common in the two sets are averaged, and second, features that are *not* in the original feature set (i), but exist in the extended set (ii), are kept with their full weight; see also Paper D.

As the method relies on a large external training set, it is suitable mainly in tasks where domain specific data can be collected, a task which can be challenging in many cases. However, preliminary results show that applying an extended feature set from one domain may also improve the classification

Method	Spoken or written	Sentiment
Classical	$0.834 - 0.927$	$0.873 - 0.912$
Neural	$0.931 - 0.973$	$0.860 - 0.965$
Our classifier	0.953	0.925

Table 5.1: The performance of various classification methods measured as accuracy over the test set in the two tasks, spoken or written text classification (Paper C), and sentiment classification of movie reviews (Paper D). The results are summarized here by giving a range of accuracies for the two categories (classical and neural) of methods against which our (linear) classifier was compared. The results are presented in more detail in the corresponding papers.

of data from a different domain.

5.2 Observations

The linear classifier trained using the method described above displays a competitive performance for the two tasks in Papers C and D. The results are summarized in Table 5.1. In both cases, the classifier outperforms the considered classical methods. Additionally, in both tasks, the classifier either outperforms or is not far below the performance of neural methods: In Paper C, the accuracy for the linear classifier is higher than that of a shallow neural network, and is less than 0.02 below that of the DistilBERT model. Similarly, in Paper D the accuracy of the linear classifier exceeds that of some neural methods (e.g., MLP and LSTM) and is not far below that of approaches relying on BERT or one of its variants. These results thus allow us to answer RQ3 positively.

In connection with Paper D, several attempts were made to improve the classifier. For example, the use of additional features, such as POS tags, was explored. Additionally, an approach for detecting and adjusting the weight of sentences that provide context (such as plot descriptions) rather than opinions was explored. An additional multiplicative weight was introduced for each sentence. For the context sentences this weight reduced their impact on the classification. However, these different approaches either had no impact on the output or displayed only a small improvement in the classification of the reviews. Substantial improvements were only obtained when applying the feature set extension in Paper D: The initial accuracy of our linear classifier was 0.8948, increasing to 0.9247 after applying the extension approach (described above in Section 5.1.3). These findings indicate a positive answer to RQ4. By applying the method from Paper D, the coverage ratio (i.e., the actual number of feature instances divided by the theoretical upper limit; see Paper D) of a text item increases, as also seen for the sentence in

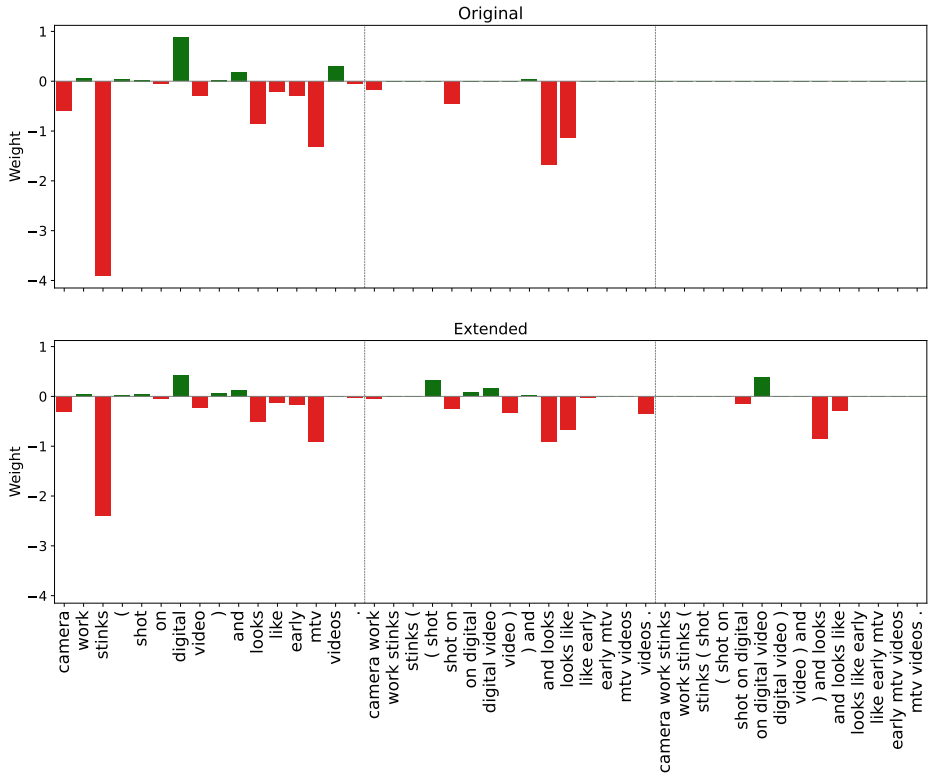

*Figure 5.2: A sentence that initially had a strong negative polarity of about -9.46, which was adjusted to -6.92 after the feature set extension. This sentence had a strong contribution in the initial (negative) classification of the otherwise positive review. The full review, with the exemplified sentence shown in bold, is as follows: "*Hilarious film about divine retribution. **Camera work stinks (shot on digital video) and looks like early MTV videos.** Turn the other cheek by looking past the visual and concentrate on the story. Laughs galore for those with a well-developed sense of irony.*" The weights of the other sentences were also adjusted such that the review was correctly classified as positive after the extension method was applied.*

Fig. 5.2.

The primary aim in Papers C and D was to provide an interpretable approach for text classification with a competitive accuracy, rather than to directly compete with the performance of DNN-based methods. The classifiers introduced in Papers C and D are human-interpretable, such that the exact impact of each n-gram feature on the classification output can be visualized. In Paper C, this is demonstrated with a tool for visualization (see also Fig. 5.1). A different visualization was used in Paper D, exemplified in Fig. 5.2.

Despite their interpretable nature, the impact of n-gram features can sometimes be counterintuitive. As each feature in the feature set is associated

with a weight, one might perhaps expect the weights of antonymous words to have similar magnitudes. This is, however, not always the case, due to the varying usage frequency of different words. In an ideal case, these weights could be aligned such that replacing a word with its antonym in a simple sentence would reverse the text sentiment, e.g., in the sentence *this movie was [great/terrible]*. To some extent, this is what happens when extending the feature set in Paper D. For example, before including the weights of the extended feature set, the weights for the unigrams *great* and *terrible* were around 1.49 and -2.24, respectively. After the extension was applied, the new weights for these unigrams were updated to approximately 1.06 and -1.56, respectively, which is slightly more balanced (though there still remains a difference).

Additionally, some (unigram) features, such as those consisting only of function words, should be associated with a small weight as they are often mostly neutral. In the preprocessing step, the removal of neutral words (such as stop words) was not applied, meaning that the feature set contained some features that could be considered neutral. Instead, their weights were adjusted in the optimization. As one of the experiments done in connection with Paper D, the features with weights close to 0 (i.e., nonpolar features) were also removed. This, however, did not improve the classification. Another aspect to consider is the weight of unigram features that consist of qualifiers (e.g., *very*, *less*, etc.) and negations (e.g., *not*, *never*, etc.). Instead of associating them with a fixed weight, one could instead consider using them as flexible features that modify the weight of the features related to their surrounding context. For example, the weight sum of a text can be increased or decreased with qualifiers (*this was very good* or *it was a bit boring*) whereas negations should often completely invert the polarity of the text.

Chapter 6

Conclusion and future work

This chapter concludes the thesis by revisiting the research questions listed in Section 1.1. Directions for future work, mainly concerning text classification, are also considered below.

6.1 Conclusion

This thesis studies interpretable (glass-box) methods in two NLP tasks, namely POS tagging and text classification. In regards to the performance of POS taggers (RQ1), the findings in Papers A and B (also discussed in Chapter 4) indicate that the accuracy of standard taggers drops considerably when they are applied over more challenging data sets, such as the ones introduced in Papers A and B (see also Section 3.1). For improving POS tagging (RQ2), a rule-based corrective approach was considered (Paper B; see also Chapter 4) that improved the POS tagger performance to some degree. These findings, however, should be seen as a proof-of-concept rather than a definitive answer to the research questions, as the size of the data sets considered in Papers A and B is rather small.

Regarding the possibility of improving the performance of linear text classifiers (RQ3), the results in Papers C and D (see also Chapter 5) show that the performance of a linear text classifier *can* indeed be improved such that the performance gap, relative to DNN-based methods, is narrowed. A specific method for improving the classifier performance, where the feature set of a text classifier was extended (RQ4), was implemented in Paper D (also described in Chapter 5). The results of this method (Paper D and Chapter 5) showed a considerable improvement in accuracy over a data set taken from the same domain as the external training set (see also Section 3.1).

6.2 Future work

The extension method in Paper D was applied in a case where a large extended data set from the same domain could be collected. As a next step, the large data set can be further extended to include samples from multiple domains. The method can then be evaluated over a broader set of examples, involving, for instance, smaller data sets where a domain-specific extended data set is difficult to generate.

Further improvements to the classification method can also be explored, for example by evaluating various feature selection methods. Another approach for improving the classifier is to use a linguistic knowledge base in the task. For example, token similarity scores can be collected from a neural network and mapped to a more interpretable representation, such as a dictionary of similar tokens (involving either single words or sequences of words, i.e., phrases). Such a dictionary of lexical items can then be used to map synonymous tokens to their counterparts in cases where a token under consideration is missing from the feature set.

Another potential research topic is to apply the methods described here in applications that deal with high-stakes decision-making where an interpretable approach is essential. In particular, the method for text classification could be used in applications related to healthcare, for example, self-harm prevention via analysis of clinical notes and patient summaries. Interpretable text classification is likely to be useful also in applications involving education and pedagogy. As an example, educators may (when desired and appropriate) use various tools to detect text generated by an AI system in student assignments, a task that is very challenging. Thus, being able to provide the *reasons* for a classification may, in this case, be valuable from an educational perspective.

Chapter 7

Summary of included papers

This thesis consists of four papers that explore the use of glass-box models in two NLP tasks. The work in Paper A introduced an initial problem in POS tagging, which motivated the extended work in Paper B, where a corrective approach for POS tagging was considered. An interpretable text classification method was introduced and applied in a classification task in Paper C. The method was also used to classify text sentiment in Paper D, where the results were further improved by expanding the coverage of potential features in the classifier.

7.1 Paper A

In Paper A, a new data set was collected for the purpose of testing existing POS taggers. Four standard POS taggers were used in the selection of the data. The reported accuracy of the standard POS taggers is typically quite high, around 0.97. When applied over the new test set, a considerable drop in tagger performance was observed: The accuracy of the standard taggers was found to be in the range $0.47-0.67$. Furthermore, a DNN-based tagger which was *not* used in the data selection process reached an accuracy of only around 0.87 over the same set, far below its reported performance. These results indicate that there are still significant challenges regarding POS tagging.

7.2 Paper B

The work in Paper A was continued in Paper B, where new sentences were generated and fully annotated, so that they were also suitable to use for training POS taggers. Here, sentences with any ambiguities were discarded.

The accuracy of the selected standard taggers over the new sentences was around $0.48 - 0.84$, and around 0.95 for the DNN-based tagger. An approach for automatically generating a set of corrective rules was proposed and implemented using the new data set. The rule-based approach for correcting the output of any tagger showed improvements (albeit rather small) in the accuracy and serves as a proof-of-concept of the proposed method.

7.3 Paper C

Classifying sentences as spoken transcripts or written text was considered in Paper C. First, a large data set of labeled sentences was generated. The classification task was carried out with a linear classifier, which was trained with a novel method introduced in the paper. The accuracy obtained by the classifier of around 0.95 is better than that obtained with other classical methods and is also comparable to that of DNN-based methods. The results indicate that the supposed trade-off between performance and interpretability can be alleviated, or possibly even eliminated, by improving glass-box methods. In connection with the paper, an tool was built and made publicly available online to test the classifier and to visualize the features that determine the class assignment for a given text.

7.4 Paper D

The classifier from Paper C was used in Paper D for sentiment classification of movie reviews. The method was enhanced by extending the feature set using a large additional data set in the same domain (movie reviews). The approach showed substantial improvements in the classifier performance: Before applying the extension method, the linear classifier reached a test accuracy of 0.8948, and with the extension, the best test accuracy increased to 0.9247. The performance of the classifier is comparable to that of pretrained DNNs, while maintaining full interpretability.

Bibliography

1. Apidianaki, M. From word types to tokens and back: A survey of approaches to word meaning representation and interpretation. *Computational Linguistics* **49,** 465–523 (2023).
2. Athota, L., Shukla, V. K., Pandey, N. & Rana, A. *Chatbot for Healthcare System Using Artificial Intelligence* in *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)* (2020), 619–622.
3. Benedetto, I., La Quatra, M. & Cagliero, L. LegItBART: a summarization model for Italian legal documents. *Artificial Intelligence and Law,* 1–31 (2025).
4. Benjamin, M. *Hard numbers: Language exclusion in computational linguistics and natural language processing* in *Proceedings of the LREC 2018 Workshop "CCURL2018–Sustaining Knowledge Diversity in the Digital Age* (2018), 13–18.
5. Berge, G. T., Granmo, O.-C., Tveit, T. O., Goodwin, M., Jiao, L. & Matheussen, B. V. Using the Tsetlin Machine to Learn Human-Interpretable Rules for High-Accuracy Text Categorization With Medical Applications. *IEEE Access* **7,** 115134–115146 (2019).
6. Berryhill, J., Heang, K. K., Clogher, R. & McBride, K. Hello, World: Artificial intelligence and its use in the public sector. *OECD Working Papers on Public Governance,* 1–184 (2019).
7. Birjali, M., Kasri, M. & Beni-Hssane, A. A comprehensive survey on sentiment analysis: Approaches, challenges and trends. *Knowledge-Based Systems* **226,** 107134 (2021).
8. Bostrom, K. & Durrett, G. Byte pair encoding is suboptimal for language model pretraining. *arXiv preprint arXiv:2004.03720* (2020).

9. Brauwers, G. & Frasincar, F. A survey on aspect-based sentiment classification. *ACM Computing Surveys* **55,** 1–37 (2022).

10. Cambria, E., Poria, S., Gelbukh, A. & Thelwall, M. Sentiment analysis is a big suitcase. *IEEE Intelligent Systems* **32,** 74–80 (2017).

11. Carabantes, M. Black-box artificial intelligence: an epistemological and critical analysis. *AI & society* **35,** 309–317 (2020).

12. Castro, J., Gómez, D. & Tejada, J. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research* **36,** 1726–1730 (2009).

13. Chai, C. P. Comparison of text preprocessing methods. *Natural Language Engineering* **29,** 509–553 (2023).

14. Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q. & Xie, X. A Survey on Evaluation of Large Language Models. *ACM Trans. Intell. Syst. Technol.* **15** (Mar. 2024).

15. Chen, H., Liu, X., Yin, D. & Tang, J. A Survey on Dialogue Systems: Recent Advances and New Frontiers. *SIGKDD Explor. Newsl.* **19,** 25–35 (Nov. 2017).

16. Cordeiro, F. R. & Carneiro, G. *A survey on deep learning with noisy labels: How to train your model when you cannot trust on the annotations?* in *2020 33rd SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)* (2020), 9–16.

17. Demner-Fushman, D., Chapman, W. W. & McDonald, C. J. What can natural language processing do for clinical decision support? *Journal of Biomedical Informatics* **42.** Biomedical Natural Language Processing, 760–772 (2009).

18. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (eds Burstein, J., Doran, C. & Solorio, T.) (Association for Computational Linguistics, Minneapolis, Minnesota, June 2019), 4171–4186.

19. Enholm, I. M., Papagiannidis, E., Mikalef, P. & Krogstie, J. Artificial intelligence and business value: A literature review. *Information Systems Frontiers* **24,** 1709–1734 (2022).

20. Fitria, T. N. Grammarly as AI-powered English writing assistant: Students' alternative for writing English. *Metathesis: Journal of English Language, Literature, and Teaching* **5,** 65–78 (2021).

21. Francis, W. N. & Kucera, H. *Brown Corpus Manual* tech. rep. (Department of Linguistics, Brown University, Providence, Rhode Island, US, 1979).

22. Giesbrecht, E. & Evert, S. *Is part-of-speech tagging a solved task? An evaluation of POS taggers for the German web as corpus* in *Proceedings of the fifth Web as Corpus workshop* (2009), 27–35.

23. HaCohen-Kerner, Y., Miller, D. & Yigal, Y. The influence of preprocessing on text classification using a bag-of-words representation. *PloS one* **15,** e0232525 (2020).

24. Klemen, M., Krsnik, L. & Robnik-Šikonja, M. Enhancing deep neural networks with morphological information. *Natural Language Engineering* **29,** 360–385 (2023).

25. Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L. & Brown, D. Text classification algorithms: A survey. *Information* **10,** 150 (2019).

26. Krishna, S., Han, T., Gu, A., Wu, S., Jabbari, S. & Lakkaraju, H. The disagreement problem in explainable machine learning: A practitioner's perspective. *arXiv preprint arXiv:2202.01602* (2022).

27. Lai, J., Gan, W., Wu, J., Qi, Z. & Yu, P. S. Large language models in law: A survey. *AI Open* **5,** 181–196 (2024).

28. Lauriola, I., Lavelli, A. & Aiolli, F. An introduction to Deep Learning in Natural Language Processing: Models, techniques, and tools. *Neurocomputing* **470,** 443–456 (2022).

29. Li, B., Qi, P., Liu, B., Di, S., Liu, J., Pei, J., Yi, J. & Zhou, B. Trustworthy AI: From principles to practices. *ACM Computing Surveys* **55,** 1–46 (2023).

30. Liu, J., Min, S., Zettlemoyer, L., Choi, Y. & Hajishirzi, H. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens. *arXiv preprint arXiv:2401.17377* (2024).

31. Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y. & Potts, C. *Learning Word Vectors for Sentiment Analysis* in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (Association for Computational Linguistics, Portland, Oregon, USA, 2011), 142–150.

32. Magar, I. & Schwartz, R. Data contamination: From memorization to exploitation. *arXiv preprint arXiv:2203.08242* (2022).

33. Maleki, N., Padmanabhan, B. & Dutta, K. *AI Hallucinations: A Misnomer Worth Clarifying* in *2024 IEEE Conference on Artificial Intelligence (CAI)* (2024), 133–138.

34. Manning, C. D. *Part-of-speech tagging from 97% to 100%: is it time for some linguistics?* in *International conference on intelligent text processing and computational linguistics* (2011), 171–189.

35. Marcus, M. P., Santorini, B., Marcinkiewicz, M. A. & Taylor, A. Treebank-3. *Linguistic Data Consortium, Philadelphia* (1999).

36. McGregor, S. Preventing Repeated Real World AI Failures by Cataloging Incidents: The AI Incident Database. *Proceedings of the AAAI Conference on Artificial Intelligence* **35,** 15458–15463 (2021).

37. Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M. & Gao, J. Deep Learning–based Text Classification: A Comprehensive Review. *ACM Comput. Surv.* **54** (Apr. 2021).

38. Molnar, C. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable* 3rd ed. (2025).

39. Navigli, R., Conia, S. & Ross, B. Biases in Large Language Models: Origins, Inventory, and Discussion. *J. Data and Information Quality* **15** (June 2023).

40. Northcutt, C. G., Athalye, A. & Mueller, J. Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv preprint arXiv:2103.14749* (2021).

41. Perikos, I., Kardakis, S. & Hatzilygeroudis, I. Sentiment analysis using novel and interpretable architectures of Hidden Markov Models. *Knowledge-Based Systems* **229,** 107332 (2021).

42. Perković, G., Drobnjak, A. & Botički, I. *Hallucinations in LLMs: Understanding and Addressing Challenges* in *2024 47th MIPRO ICT and Electronics Convention (MIPRO)* (2024), 2084–2088.

43. Petrov, S., Das, D. & McDonald, R. *A Universal Part-of-Speech Tagset* in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)* (European Language Resources Association (ELRA), Istanbul, Turkey, May 2012), 2089–2096.

44. Ribeiro, M. T., Singh, S. & Guestrin, C. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier* in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Association for Computing Machinery, San Francisco, California, USA, 2016), 1135–1144.

45. Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L. & Zhong, C. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistic Surveys* **16,** 1–85 (2022).

46. Schuster, M. & Nakajima, K. *Japanese and Korean voice search* in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2012), 5149–5152.

47. Sennrich, R., Haddow, B. & Birch, A. *Neural Machine Translation of Rare Words with Subword Units* in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (eds Erk, K. & Smith, N. A.) (Association for Computational Linguistics, Berlin, Germany, 2016), 1715–1725.

48. Simonyan, K., Vedaldi, A. & Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).

49. Slack, D., Hilgard, S., Jia, E., Singh, S. & Lakkaraju, H. *Fooling lime and shap: Adversarial attacks on post hoc explanation methods* in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society* (2020), 180–186.

50. Templeton, A., Conerly, T., Marcus, J., Lindsey, J., Bricken, T., Chen, B., Pearce, A., Citro, C., Ameisen, E., Jones, A., Cunningham, H., Turner, N. L., McDougall, C., MacDiarmid, M., Freeman, C. D., Sumers, T. R., Rees, E., Batson, J., Jermyn, A., Carter, S., Olah, C. & Henighan, T. Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet. *Transformer Circuits Thread* (2024).

51. Von Eschenbach, W. J. Transparency and the black box problem: Why we do not trust AI. *Philosophy & Technology* **34,** 1607–1622 (2021).

52. Wang, H., Wu, H., He, Z., Huang, L. & Church, K. W. Progress in Machine Translation. *Engineering* **18,** 143–153 (2022).

53. Zhang, Y., Jin, H., Meng, D., Wang, J. & Tan, J. A comprehensive survey on process-oriented automatic text summarization with exploration of LLM-based methods. *arXiv preprint arXiv:2403.02901* (2025).

54. Zhou, H., Zhang, Y., Li, Z. & Zhang, M. *Is POS Tagging Necessary or Even Helpful for Neural Dependency Parsing?* in *Natural Language Processing and Chinese Computing* (eds Zhu, X., Zhang, M., Hong, Y. & He, R.) (Springer International Publishing, Cham, 2020), 179–191.