Universität Zürich[UZH]

Master's thesis
for the degree of
Master of Arts UZH

Lemma Disambiguation in Multilingual Parallel Corpora

Author: Jasmin Heierli
Student ID: 07-741-432

Examiner: Prof. Martin Volk
Supervisor: Dr. Simon Clematide
Institute of Computational Linguistics

Submission date: 18.06.2018

Abstract

We present novel and efficient approaches to lemmatisation by improving and building on the output of standard lemmatisation tools such as the TreeTagger. Lemmatisation is the process of reducing an inflected word form to its base form and it can be key in the process of reducing sparse data for morphologically rich languages, such as Finnish or German, because highly inflected word forms are relatively rare. A rich morphology also leads to more ambiguities. These ambiguities in context usually do not pose a problem to humans, but they pose a problem to computers. In computational linguistics lemmatisation is performed automatically and often relies on a lexicon known to the system. Ideally each word form is in the lexicon and assigned with a single lemma. However, this is often not the case, as the system is confronted with unseen word forms in a new data set, the lexicon contains erroneous entries or the same word form occurs with more than one lemma candidate because it is ambiguous. The TreeTagger is such a system and a standard POS-tagging tool that can output lemmas to word forms that are contained in the lexicon.

We develop two machine learning approaches for lemma disambiguation in German, French, Italian and Finnish that yield better overall performance than pre-existing quantitative approaches. We apply our approaches to the *Full European Parliament Corpus v. 6*, a multilingual, multiparallel corpus. The first approach is a distant-semi supervised approach, which is trained on the unambiguous lemmatisations of lemma candidates of ambiguous lemmas. This approach does not need any manually annotated data for training. The main disadvantage is the low coverage, as a large amount of ambiguous lemmas do not occur unambiguously in the corpus. Therefore, we also developed an active learning pipeline that lets users select the correct lemma in several carefully chosen examples of ambiguous lemmas with lack of evidence for lemma candidates. We draw the observations for manual annotation from precomputed clusters, which help to find minority classes and outliers. The manually disambiguated data is used for a supervised machine learning approach.

For the evaluation of the two approaches, we created gold standards for all four languages. The larger part A is for evaluating the distant semi-supervised approach and the smaller part B is for evaluating the active learning pipeline. The distant semi-supervised approach yields results of above 85 % precision, which is above the majority class baseline and exceeds a previously developed quantitative approach applied on the same corpus in recall and F_1-score. The results for the active learning pipeline are equally promising with a precision above 90 % for German.

Acknowledgement

First of all, I would like to thank my supervisor, Dr. Simon Clematide, for introducing the topic for this thesis to me and being encouraging and patient until its completion. His door was always open and he always tried to squeeze in some time for answering my most urgent questions and spent extra time on reading and improving my work. I also want to thank Johannes Graën for providing me with the data of his former analysis and his knowledge and expertise with the corpus. He also supported me with issues regarding the server infrastructure and I appreciated his inputs in our after-lunch discussion. I am also grateful for Phillip Ströbel's proofreading and his introduction to Multivec and word embeddings, which took my research another step further. I would also like to thank the rest of the staff from the institute of Computational Linguistics for the warm and welcoming atmosphere and the lunch times we have spent together.

My special thanks go to Chiara Baffelli, Michela Rossi and Vanessa Kasper, who diligently and free of charge helped me with the Italian and Finnish gold standards. Without your precious work, I could not have conducted my experiments in the manner as they are. I also thank Sara Wick for proofreading and the encouragement throughout.

Last, but not least, I am grateful for my family's patience and support. I owe many thanks to my mother for the many hours of additional babysitting, as well as our childminder, who took on extra shifts. I also thank my children, Olivia and Vivien, for their extra patience and the bearing of my absences. Finally, without the support and encouragement of my husband, this project could not have been accomplished. Thank you for your unconditional love and support, Thomas.

Contents

List of Figures

List of Tables

List of Acronyms

CBOW	continuous bag-of-words
CCC	Cophenetic Correlation Coefficient
CL	Computational Linguistics
CRF	Conditional random field
DE	German
FEP6	Full European Parliament Corpus version 6.0
FI	Finnish
FR	French
GTB	Gradient Tree Boosting
IT	Italian
NER	Named Entity Recognition
NLP	Natural Language Processing
NLTK	Natural Language Processing Toolkit
OCR	Optical Character Recognition
POS	Part-Of-Speech
SVM	Support Vector Machines
TT	TreeTagger
WSD	Word Sense Disambiguation

1 Introduction

1.1 Motivation

A thorough and meticulous annotation is the foundation for most natural language processing (NLP) applications. A subtask of linguistic annotation is lemmatisation, which is key to numerous NLP tasks, such as parsing, machine translation, ontology creation, lexicographical tasks, information retrieval and word sense disambiguation (WSD).

Lemmatisation is the process of reducing an inflected word form to its 'lemma', the base form. The concept of 'lemma' has different meanings, depending on the field of study.

- **Linguistics** A 'lemma' is considered as keyword in a dictionary. It represents a lexeme, which subsumes all word forms of the word paradigm of a word in a certain language (Glück and Rödel, 2016, p. 96).
- **Computational Linguistics** A 'lemma' is the normalised base form of all word forms that belong to the same word, having the same stem, part-of-speech tag and the same word-sense (Jurafsky and Martin, 2017, p. 20).

The two definitions are closely related and yet they have entirely different predispositions. A lemma in the linguistic context refers to a lexeme representation with a dictionary entry, while in the context of computational linguistics (CL) a dictionary entry is not necessarily involved. We will illustrate this with two examples. The word forms *sings*, *singing* and *sang* belong to the lemma *sing*, which you will find in any average dictionary. If we have a look at the number *11*, or a full stop *'.'*, it becomes difficult to determine a lemma in the linguistic sense. *11* and *'.'* cannot be found in most dictionaries, as there is an infinite number of digit combinations and punctuation marks do not carry lexical meaning. However, in CL, we usually lemmatise all tokens, regardless of whether they carry lexical or grammatical meaning, since many applications rely on lemmatisation and even units such as punctuation deliver crucial information. Such tokens receive a lemma identical to their word form, for tasks such as tagger training, or a normalised lemma, such as *'#'* for any digit number or a *'.'* for punctuation in order to reduce orthographic

```
Envelopes NNS envelope
and CC and
ballot NN ballot
papers NNS paper
have VHP have
been VBN be
distributed VVN distribute
. SENT .
```

Figure 1.1: A POS-tagged and lemmatised English sentence.

variation depending on purpose and lemmatisation convention. The conventions may indeed differ across languages and applications. In German and English, verbs are usually reduced to the infinitive form, such as *went* → *go* or German *ging* → *gehen*. In Latin on the other hand, verbs are reduced to the 1st person singular, such as *amo* ('I love'). Lemmatisation in CL usually follows these conventions, particularly when applications rely on additional data from lexicons or are conceptualised to extend the such.

While published lexicons are written manually, CL lemmatises words automatically. We will look at the different methods later, and will now briefly explore the obstacles and peculiarities of automatic lemmatisation. Automatic lemmatisation often relies on a lexicon known to the system. Ideally, each encountered word form is in the lexicon and unambiguously assigned to a lemma as in the tagger output of the *TreeTagger* (Schmid, 1994) in Figure 1.1, consisting of word form, part-of-speech and lemma.

However, the first obstacle is that the lexicon can contain incorrect lemma candidates, such as in 1.1.

(1.1) *stattet* *VVFIN* *statten*
word form POS tag lemma
Die Union stattet sich mit den Mitteln aus ...
'The Union is equipping itself with the means ...'

The lemma *statten* does not exist in German. The context of the word form *stattet* makes clear that the lemma has to be *ausstatten* ('equip'), as it is a particle verb and the particle *aus* follows 5 tokens later. Such errors are mainly caused by separated verb prefixes that cannot be reattached automatically by the tagger. The other obstacle lies in the nature of language. Lemmatisation lexicons are finite, while natural language is not. This is shown in the German example 1.2. *Gegenseitigkeitsvereinbarung* ('reciprocal agreement') is a compound. Compounds and named entities, such as 1.3, are frequently not covered by the lexicons as the possibilities

for novel word creations are endless.

(1.2) *Anwendungsbereichs* *NN* ⟨*unknown*⟩
word form POS tag lemma
'scope's' NN ⟨unknown⟩

(1.3) *Obamas* *NP* ⟨*unknown*⟩
word form POS tag lemma
'Obama's' NP ⟨unknown⟩

There are out of the box tools available to solve the issue with unidentified compounds and named entities. The issue of unknown compounds can be tackled by compound splitters, such as *BananaSplit* (Ott, 2006). *Bananasplit* is based on recursive look-ups in an dictionary and splits nouns, adjectives, and verbs in exactly two parts if the compound is not in the lexicon. There are also good options, such as the *Stanford Named Entity Recognizer* (Finkel et al., 2005), available to solve the issue with unknown named entities. Named entities are usually divided into different classes, such as people, locations and organizations, and then identified by a classifier, such as the aforementioned *Stanford Named Entity Recognizer* (Finkel et al., 2005), which is based on conditional random fields. This process is called named entity recognition (NER).

Lemmatisation can be key in reducing sparse data for better system performance. This is particularly crucial, when dealing with morphologically rich languages, such as Finnish or German because a considerable amount of highly inflected forms is rare. A rich morphology may not only result in an increased amount of sparse data, but also in more ambiguities. Resolving morphological ambiguity in context does not pose a problem to most humans, but it poses a huge problem to machines. Thus, some tools, such as the *TreeTagger* (TT) return more than one lemma candidate for ambiguous word forms, as shown in example 1.4.

(1.4) *Akten* *NN* *Akt|Akte*
word form POS tag lemma candidates
file.PLURAL NN 'act|file'

Given enough context, humans immediately know whether the lemma needs to be *Akte* ('file') or *Akt* ('act') because the grammatical gender of *Akt* does not agree with the case in context and the resulting meaning does not make sense. The TT on the other hand, outputs all lemmas that it has seen at some point associated with this particular word form. This is where we want to strive for novel, efficient solutions that improve and build on the output of standard lemmatisation tools such as the TreeTagger.

1.2 Research Questions

The research questions that shall be answered in this thesis, are:

1. How good is the lemmatisation quality of the TreeTagger models for German, French, Italian and Finnish? We look into their performance on the European Parliament corpus.
2. How can a distant semi-supervised machine learning approach disambiguate ambiguous lemmatisation in a multilingual multiparallel corpus set-up with monolingual context-based features, as well as multilingual features?
3. How can active learning techniques be efficiently applied in order to generate training material for a supervised machine learning approach where the distant semi-supervised approach cannot be used due to missing training data?

The first question aims at investigating the overall lemmatisation quality of the TreeTagger for German, French, Italian and Finnish. In a perfect world, our corpus would consist of tokens with exactly one correctly assigned, unambiguous lemma. However, in reality our corpus has a considerable amount of tokens with more than one or no lemma candidates. Particularly morphologically rich languages, such as the four languages in question are hard to lemmatise. Additionally, the assigned lemmas can be erroneous. Thus, the quality shall be investigated in terms of the amount of ambiguities per language, as well as incorrectly assigned lemmas. The incorrectly assigned lemmas are counted in a subsample, which is manually disambiguated in order to establish a gold standard for German, French, Italian and Finnish. The gold standard will be used to evaluate the machine learning approaches presented in question two and three below.

The second question is based on the assumption that aligned translations in a multilingual parallel corpus can be exploited in order to disambiguate lemmas. We propose the application of a distant semi-supervised machine learning approach which, gains information based on multilingual as well as intralingual features. Ambiguous lemmas in the form of *Stimme*|*Stimmen* can be split into their individual lemma candidates, such as *Stimmen* ('voices', noun plural) and *Stimmen* ('voting', noun). We search for word forms, which feature the lemmas from the split ambiguous lemma as unambiguous lemma. The unambiguous lemma becomes the label and its word form within context the training instance. and used as labelled training instances for a machine learning algorithm. As the task of lemma disambiguation is closely related to word sense disambiguation (WSD), our work will be largely inspired by the findings of WSD tasks, which have been tackled by machine learning for several years. As compared to WSD, lemma disambiguation only distinguishes between different senses if they are expressed in two different lemmas. Thus, lemma

disambiguation is more superficial and may be regarded as preliminary stage to WSD, as it reduces the amount of ambiguities on the word form level and only homonymous and polysemous lemmas should remain. However, lemma disambiguation and WSD share the difficulty of changing sets of classes, depending on the word that has to be classified, as opposed to a fixed set of classes for most classification tasks, such as document classification, tagging or named entity recognition. Additionally, our approach is novel, as it also includes intralingual features as a back-off in case that multilingual features are not available. While other approaches, such as Volk et al. (2016) and Graën (2018), solely rely on translation information, our system will also take into account monolingual context. Hence, we will like WSD assume that the monolingual and parallel multilingual context of each token will deliver the necessary information for a machine learning algorithm to take these decisions (Navigli, 2009, p. 28).

The third question addresses the problem of gaining training data for cases for which not all lemma candidates occur as unambiguous lemma in the corpus. The distant, semi-supervised machine learning approach cannot resolve these ambiguities. The amount of these ambiguous lemmas is huge and the classes are often imbalanced, which would result in a very expensive manual annotation task, as we would also need to search examples for the minority class in order to train a classifier. Thus, we will present an active learning approach that prompts the user for manual annotations. The sample for manual annotation is drawn from previously clustered data, in order to prevent bias towards the majority class and search for missed candidates. The clustering is based on multilingual word embeddings. The manually annotated occurrences will be used to apply a supervised approach that disambiguates the rest of occurrences automatically.

1.3 Thesis Structure

Chapter 2 introduces a selection of previous works on topics related to lemmatisation and lemma disambiguation. We will introduce lemmatisers and morphological analysers and have a look into two methods that specifically tackle lemma disambiguation. Furthermore, we will present WSD approaches that have an experimental set-up that is similar to ours.

The data for our experiments in presented in Chapter 3. We introduce the corpus and the data format, with which we have conducted our experiments. For the experiments we have developed a gold standard lemmatisation for ambiguous lemmas in German, French, Italian and Finnish, which we will also introduce in this chapter. Additionally, we will investigate the accuracy and coverage of the

TreeTagger on ambiguous lemmas.

We outline the the experiments with the distant semi-supervised approach in Chapter 4. We will introduce the machine learning algorithms and features that we have tested and how we obtained our training and test-set and the labels.

Chapter 5 introduces active learning and how we used it in order to disambiguate lemmas that cannot be tackled with the previously presented approach. We also introduce a method to successfully draw examples of the minority classes from unlabelled data for a supervised machine learning approach with a handful of manually labelled data.

We present the results of our experiments in Chapter 6. Firstly, we will present the results and improvements of the distant semi-supervised approach for German, French, Italian and Finnish, and, secondly, we present the results of the supervised approach based on active learning. We also evaluated the quantitative approach of Graën (2018), who provided us with his data.

In Chapter 7 we present the conclusions we made after having worked on this topic. We will discuss the benefits and drawbacks of our approaches and we will suggest ideas and inspiration for future experiments.

2 Previous Works

2.1 Morphological Analysis and Lemmatisation

Recent findings show that it is favourable to apply models with joint lemmatisation and morphological tagging. In the following section, we will introduce three systems with language independent joint lemmatisation and morphological tagging, which do not rely on external lexical recourses or morphological dictionaries and analyzers.

Morfette is a morphological tagger and lemmatiser developed by Chrupala et al. (2008), which is based on a Maximum Entropy classifiers for each task and a decoding module that matches the best output sequences of the two classifiers for a sequence of words. Their system is language-independent and also copes well with morphologically rich languages with sparse word forms (Chrupala et al., 2008, p. 2362). In order to deal with sparse, unseen word forms and lemmas, Morfette's lemmatisation model is based on lemma induction. Each lemma is induced by a shortest edit script, which transforms a string into its lemma (Chrupala et al., 2008, p. 2362). If we have the German form *schreit* ('scream', 3rd person, singular, present) the following edits are necessary to transform it into its lemma *schreien* ('scream'): $\{\langle D, t, 7\rangle, \langle I, e, 8\rangle, \langle I, n, 9\rangle\}$, where a triple stands for delete/insert character at a certain position. Each string is reversed, as most inflection is expressed through suffixation and the shortest edit scripts should be as independent from word length as possible. These edit scripts are assigned as classes to word form lemma pairs. Both, lemmatisation and morphological tagging are based on a number of arbitrary and context-based features, which can be easily extended and refined for specific languages and domains. The two Maximum Entropy Models return probability distributions for all possible morphological tags given the context and all possible lemma classes given the context and morphological tag. A beam search algorithm then combines the n sequences with the highest probability of morphological tags and lemmas for a given word form. They report a considerable error reduction rate around 60 %, for Morfette against their baseline system, consisting of a memory-based tagger generator (Chrupala et al., 2008, p. 2365). Additionally, models for lemmatisation and morphological tagging can be trained separately and even with different algorithms, which is a plus, if training corpora with both annotations are

unavailable for training (Chrupala et al., 2008, p. 2366)

Müller et al. (2015) developed a log-linear model called LEMMING that models lemmatisation and morphological analysis at the same time. Their system operates on token level and is also able to lemmatise unknown word forms, due to the idea that lemmatisation is a"string to string transduction task" (Müller et al., 2015, p. 2269). This means that they translate word forms, such as *worked* character by character – w-w, o-o, r-r, k-k, e-ø, d-ø – into its lemma *work*. These transformations are exploited as features for the logistic regression model, alongside irregular word forms, possible prefixes and suffixes, and edit trees. The edit trees contain all possible affixation or deletions a lemma may accept in order to form affiliated word forms (Müller et al., 2015, p. 2269). They linked all these features with the POS and morphological attributes, which are expressed through these features (Müller et al., 2015, p. 2270). This feature set also enables the model to lemmatise unknown word forms. The final model is a tree-structured conditional random fields (CRF) model that combines the previously mentioned lemmatisation with a higher-order CRF for the sequence modelling of morphological tags, as depicted in Figure 2.1.

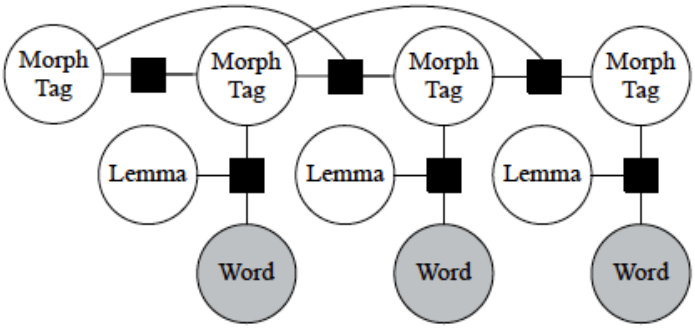

Figure 2.1: Illustration of the tree structured CRF model as presented in Müller et al. (2015)

LEMMING requires a corpus with gold standard tags and lemmas for training, and does not require more training data than a state of the art POS tagger, according to Müller et al. (2015, p. 2271). As the model is token-based and does not rely on morphological dictionaries or analyzers, it can be trained on any gold standard tagged and lemmatised corpus.

The German morphological analyser *GerTwol* by Haapalainen and Majorin (1994) is based on an entirely different approach. They developed a finite-state transducer based system that analyses and lemmatises German word forms. The word form lexicon contains 85,000 word forms, of which the majority is from *The*

```
"Abgabebestimmungen"
Ab|gab~en#be|stimm~ung S FEM, PL NOM
Ab|gab~en#be|stimm~ung S FEM PL AKK
Ab|gab~en#be|stimm~ung S FEM PL DAT
Ab|gab~en#be|stimm~ung S FEM PL GEN
```

Figure 2.2: Sample output of GerTwol for the German word *Abgabenbestimmungen* ('charges regulations').

Collins German Dictionary[1] with additional nouns and proper names (Haapalainen and Majorin, 1994). However, the number is not absolute because GerTwol is also equipped with a large set of derivational morphological and compounding rules, which enables the analysis of new compounds as well as nominalised adjectives and verbs. The analysis works on two levels. The first level is the lexical level and the second level is the surface level and each string on the lexical level has a corresponding string on the surface level (Haapalainen and Majorin, 1994). The output of the analysis consists of all possible base forms and morphological analyses, as GerTwol does not consider context. Thus, it is unsuitable as a sole analyser, but works well in combination with a POS tagger or for the extension of POS tagger lexicons. An analysis can contain strong boundaries, marked by '#' that separate independent words, soft word boundaries, marked by '|' that separate dependent morphemes or dashes that separate inflectional or derivational morphemes (Haapalainen and Majorin, 1994). Figure 2.2 shows a sample output of the German word *Abgabenbestimmungen* ('charges regulations'). GerTwol correctly analyses the compound consisting of *Abgabe* ('charges') and *Bestimmung* ('regulation'). Additionally the soft boundaries indicate that *Gabe* ('gift') and *Stimmung* ('mood') are also valid German morphemes, but attached to dependent morphemes. The second column is the morphological analysis, the POS followed by other morphological features. Eventually, GerTwol is a good addition to POS taggers for German, as it is very fast, contains the essential German vocabulary, which is extended by compounding and derivational morphological rules, and detects ambiguous word forms.

2.2 Neural Lemmatisation

One of the most recent advances in the field of lemmatisation is *Lematus* by Bergmanis and Goldwater (2018). Lematus is a context-sensitive lemmatiser, which solely

[1] *The Collins German Dictionary*, revised edition 1991, Copyright HarperCollins Publishers.

performs lemmatisation without any additional information, such as morphological tagging or POS tags, apart from character-level context (Bergmanis and Goldwater, 2018, p. 1391). Their model is built with the *Nematus* toolkit by Sennrich et al. (2017) and is a "deep attentional encoder-decoder with a 2-layer bidirectional encoder with a gated recurrent unit (GRU) and a 2-layer decoder with a conditional GRU in the first layer followed by a GRU in the second layer" (Bergmanis and Goldwater, 2018, p. 1393)[2]. As the model of Sennrich et al. (2017) is geared towards machine translation and not monolingual lemmatisation, they adapted some of the model parameters to fit their purpose. They trained models for 20 different languages on the Universal Dependencies Treebank 2.0[3]. The input sequence of their model is a space-separated character sequence of a token and all the characters before and after this token in a predefined context window of the size N. Figure 2.3 shows an example for the Latvian word ceļu, which is the genitive plural of ceļš ('road'), with a context window of size $N = 15$. The `<s>` mark token boundaries, whereas `<lc>` and `<rc>` mark left and right context respectively.

```
s a k a <s> p a š v a l d ī b u
        <lc> c e ļ u <rc>
u n <s> i e l u <s> r e ģ i s t r
```

Figure 2.3: Illustration of the input token *ceļu* for Lematus with the character-based context in a context window of $N = 15$, as in Bergmanis and Goldwater (2018, p. 1393)

They tested several models with a character-based context between 0 (only the token itself) and 25. They measured their models against Morfette, LEMMING and another neural network based system as well as a majority baseline, for which they took the lemma that has been seen most frequently or first with the word form in question (Bergmanis and Goldwater, 2018, p. 1395). Bergmanis and Goldwater (2018, p. 1395) found that a character-based input with a context window of $N = 20$ performs best for all languages against Morfette and the other neural lemmatisation approach, and for 15 languages against LEMMING. However, they noticed that the character-based model performs better on ambiguous lemmas, while the context-free model performs better on unseen lemmas. Additionally, they found that ambiguity and unseen tokens negatively correlate in the 20 languages that they investigated. A given language either expresses morphosyntactic functions through distinct word

[2]For a detailed description of the architecture, see Sennrich et al. (2017)

[3]`https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1983` (Accessed 4 May 2018).

forms, leading to a higher productivity and more unseen word forms, or non-distinct word forms, leading to more ambiguous forms (Bergmanis and Goldwater, 2018, p. 1394). Thus, the model may be chosen according to the productivity of a given language. Said ambiguous word forms are only resolved with Lematus, if they are resolved in the training corpus, i.e. the Universal Dependencies Treebank 2.0. However, German is lemmatised with the TreeTagger, which returns pipe-separated lemma candidates for ambiguous word forms. A good performance on ambiguous lemmas for Lematus, thus means reproducing such pipe separated lemma pairs.

2.3 Lemma Disambiguation Approaches

The two lemma disambiguation approaches presented in this section are relevant, as they firstly try to select the correct lemma out of two or more lemma candidates, and secondly also exploit multilingual parallel corpora.

Volk et al. (2016) present a purely quantitative approach, which exploits parallel data for disambiguating ambiguous lemma. They use parallel texts in English and German from the *Credit Suisse Bulletin Corpus* in order to resolve the lemma ambiguities in the full corpus, which also includes monolingual German texts. The TreeTagger (TT) outputs a considerable amount of lemmas with more than one candidate, such as the lemmas *Dose* and *Dosis* for the word form *Dosen* (Volk et al., 2016, p. 293). However, the English translation often reveals the intended lemma of the token in question, as *Dose* translates to *can* and *Dosis* to *dose*. Thus, they extracted all German sentences that contain an ambiguous word form with their corresponding English sentences and searched for the most probable lexical translation via word alignments (Volk et al., 2016, p. 293). Then, they compared whether the translation probability of the English lemmas is higher to one of the German candidates, and if so, the option with the higher probability was accepted as the resolution. On a sample of 100 lemmas, they reached a precision of 0.97. However, only 16 % of all lemmas could be resolved, which results in a F_1-score of 27 %[4].They could improve their recall to 75 % by accepting lemmas as unambiguous candidate, when they occur as sole possible translation for this word form in the English lemma alignments (Volk et al., 2016, p. 293). This measurement slightly reduced the precision to 0.93 %, which leads to an F_1-score of 83 %. The recall was further increased by lowercasing the German lemmas in order to perform a case-insensitive search, which would specifically tackle nominalised verb forms[5]. This results in a recall of

[4]We calculated and added the F_1-scores, as they are not provided in the paper.

[5]They hope to find evidence for the lemma *spielen* ('to play' in the corpus, if they cannot find *Spielen* ('playing').

84 % and a precision of 91 % with an F_1-score of 83 %. They did not test their method for other language pairs despite the promising results.

Graën (2018) seized on the promising results of Volk et al. (2016) and enhanced their approach by adding more translations from all aligned languages in the *Full European Parliament Corpus v.9*. He computes the global distribution matrix of all lemmas. However, he only considers optimal word alignments, for which all four word aligners agree in both directions (Graën, 2018, p. 31). Then he checks, whether each lemma candidate of a certain ambiguous lemma occurs in the corpus. If a lemma candidate does not occur as unambiguous lemma in the corpus, he excludes it as option because he cannot not calculate its probability. In these cases, he selects the lemma candidate that has unambiguous evidence in the corpus (Graën, 2018, p. 32). In a next step, he searches for corresponding tokens in the optimal alignments and looks up the alignment probability of their lemmas to each of the lemma candidates if applicable (Graën, 2018, p. 31). Eventually, he chooses the lemma candidate with the highest overall probability as sole replacement with the equation in 2.1.

(2.1)

$$\lambda_s^{final} = argmax\lambda_s^i \sum_n p_a(\lambda_s^i \mid \lambda_t^n)$$

The replacement lemma candidate is calculated by the sum of the probabilities of the n lemmas from the aligned tokens (λ_t^n) given each lemma candidate(λ_s^i). This approach could not tackle cases, for which no optimal alignments exist in the corpus, or all of the optimal alignments do not have a lemma. He conducted a small evaluation, in which he found 2 errors across 53 disambiguated lemmas. However, the evaluation sample is very small. Graën (2018, p. 34) also applied this method on FEP6 and he kindly provided us with his lemma disambiguations, which we will discuss and evaluate in Chapter 4.

2.4 Word Sense Disambiguation

Dimitar Kazakov (2013) developed a multilingual approach to WSD. Word sense disambiguation is a similar problem as lemma disambiguation, as the latter eases the path to the former. They used a subpart of the Europarl corpus and applied sentence and word alignment in order to assign the same ID to the aligned tokens (Dimitar Kazakov, 2013, p. 338). Further, they stored the frequency for each translation for a certain lemma. This results in a multilingual synset for each lemma. The combinations of lemma translations then helped to derive the sense of the lemma

in question, as some translations highly correspond to a specific word sense (Dimitar Kazakov, 2013, p. 339). This strategy lead to an average ambiguity reduction of 36.45%. Additionally, they note the high dependence on word alignment of their multilingual lemma disambiguation approach.

Lefever and Hoste (2009) present a multilingual unsupervised word sense disambiguation task without the tedious task of manually sense-tagging each polysemous noun. Instead of manually tagged data, they use the Europarl corpus and set English as source language and five languages – Dutch, French, German, Spanish and Italian – as target languages that should disambiguate the source language (Lefever and Hoste, 2009, p. 82). This approach does not only rely less on manually tagged data, but it is language independently applicable. The approach is based on the hypothesis that the different word senses of a word in English are lexicalised differently in the five target languages. If two translations match the same sense it might be a hint that these two target language words are synonymous (Lefever and Hoste, 2009, p. 84). This is essentially the same hypothesis, on which our work is based. Just as they assume that different word senses are lexicalised differently in other languages, we assume that the lemma candidates of ambiguous word forms are lexicalised differently in other languages.

Ng et al. (2003) evaluate a bilingual approach to disambiguate sense-tagged nouns with automatically acquired training data from an English-Chinese parallel corpus. Their method has the advantage that no manually disambiguated data is required for the training of their algorithm, as they use the translations of the ambiguous word as "sense-tags". Their approach is very similar to Diab and Resnik (2002) who found it to be the most successful unsupervised method for WSD (Ng et al., 2003, p. 455). They also list a few problems that might occur: it is difficult to tell how large the parallel corpus needs to be, whether the word alignment accuracy suffices for their purpose and whether they can stand the pace of state-of-the-art supervised WSD approach. For their new approach they assume that every Chinese translation of an ambiguous English noun refers to a certain sense of the English noun in question. Sometimes, two ore more senses are lumped together to one sense, as they translate into the same Chinese word (Ng et al., 2003, p. 457). Nouns that are lumped into one remaining sense only, are excluded from the WSD algorithm as there is no longer a decision to be made (Ng et al., 2003, p. 458). Eventually, they state that the Naive Bayes algorithm trained on parallel English-Chinese data could probably deliver comparable data, if it was trained domain independently and all the senses were represented in the parallel corpus.

Lefever and Hoste (2014) present a language independent WSD system that exploits word alignments of four languages – French, Dutch, Italian, Spanish and

German – as sense tags with English as input language. They automatically extracted translation labels and features via word alignments, assuming that evidence from multiple languages into the feature vector is more informative than just one (Lefever and Hoste, 2014, p. 340). They hope that the automatically acquired translation can at least partially solve the data-acquisition bottleneck (Lefever and Hoste, 2014, p. 340). In order to test their system, they selected 20 polysemous nouns with at least 3 *WordNet* senses that occur at least 50 times in the *Europarl* corpus (Lefever and Hoste, 2014, p. 341). They used a sentence-aligned version of the *Europarl* corpus by Koehn (2005). For word alignment they used GIZA++ (Och and Ney, 2003).They extracted all word alignments for their 20 nouns from the corpus and had annotators cluster them manually into two-level sense clusters with the main senses on top and the other, more fine grained senses at the bottom (Lefever and Hoste, 2014, p. 347).They serve as sense labels. For the classification they used the memory-based learning (MBL) algorithm with local context features, namely cooccurring words with the target word, extracted from the English source sentence, alongside a binary bag of words feature based on the the translations of the four languages (Lefever and Hoste, 2014, p. 353). Their best model for English yields an average accuracy of 75.2 % with all features. Lastly, the developed similar for with each of the other 5 languages – French, Dutch, Italian, Spanish ans German – as input languages and the other languages as translation features. The models outperformed all winning SemEval-2 systems, apart from the model Spanish (Lefever and Hoste, 2014, p. 356). Additionally, they found that relying on translation features from fewer languages lowers their accuracy, which confirms they initial hypothesis that multilingual features outperform bilingual features.

3 Data

This chapter introduces the corpus, which we chose for our project. We explain the data format, which we used for most of our experiments and how it was produced. Then, we focus on the lemmatisation and the distribution of ambiguous lemmas across languages.

3.1 The European Parliament Corpus

We work with the *Full European Parliament Corpus v. 6* (*FEP6*) compiled and preprocessed for the *SPARCLING* project, a project of the Department of English and Computational Linguistics at the University of Zurich[1]. The original version of the *Europarl* corpus was compiled by Philipp Koehn (2005) by crawling the website of the European Parliament. The resulting corpus contains sentence-aligned texts from debates of the European Parliament over 15 years, from 1996 to 2011. The originally mostly oral debates were transcribed into grammatically correct written text and later translated into all official languages. However, the documents are not necessarily translated directly from the source language into all 22 languages, as it would require many translators with different translation pair competences. Hence, they used a pivot system, in which they translated the documents first into the four most used languages and then into the remaining less spoken languages (Parliament, 2008). The resulting transcriptions from the debates of the European Parliament are divided into different chapters, each covering one topic. The different chapters may cover votings, debates, as well as formerly written statements, read to the other members of the parliament.

Unfortunately, this large collection of parallel texts contains a large number of errors, which could have a negative impact on further processing steps, such as PoS tagging or, parsing and turn alignment, and applications that rely on their output. Graën et al. (2014) found inconsistencies with character encoding, when alternatives were available, incompletenesses of speaker information extracted from the web or direct incorporations of them into the speaker's turn. Additionally, Koehn applied

[1] `http://www.cl.uzh.ch/de/research/completed-research/sparcling.html` (accessed 11 December 2017).

```
235444658 In in ADP APPR _ 235444661 PP ES 132423 834115 9700618 es 1550 BADIA_I_CUTCHET__maria _
235444659 dieser dies PRON PDAT _ 235444660 DET ES 132423 834115 9700618 es 1550 BADIA_I_CUTCHET__maria _
235444660 Hinsicht Hinsicht NOUN NN _ 235444658 PN ES 132423 834115 9700618 es 1550 BADIA_I_CUTCHET__maria 235445040|235445411
235444661 möchte mögen VERB VMFIN _ _ ROOT ES 132423 834115 9700618 es 1550 BADIA_I_CUTCHET__maria 235445045|235445046|235445417
235444662 ich ich PRON PPER _ 235444661 SUBJ ES 132423 834115 9700618 es 1550 BADIA_I_CUTCHET__maria _
235444663 zunächst zunächst ADV ADV _ 235444670 ADV ES 132423 834115 9700618 es 1550 BADIA_I_CUTCHET__maria 235445042|235445415
235444664 eine eine DET ART _ 235444666 DET ES 132423 834115 9700618 es 1550 BADIA_I_CUTCHET__maria _
235444665 kurze kurz ADJ ADJA _ 235444666 ATTR ES 132423 834115 9700618 es 1550 BADIA_I_CUTCHET__maria 235445050|235445420
235444666 Anmerkung Anmerkung NOUN NN _ 235444670 OBJA ES 132423 834115 9700618 es 1550 BADIA_I_CUTCHET__maria 235445051|235445421
235444667 zur zu ADP APPRART _ 235444666 PP ES 132423 834115 9700618 es 1550 BADIA_I_CUTCHET__maria _
235444668 frühkindlichen frühkindlich ADJ ADJA _ 235444669 ATTR ES 132423 834115 9700618 es 1550 BADIA_I_CUTCHET__maria 235445053
235444669 Bildung Bildung NOUN NN _ 235444667 PN ES 132423 834115 9700618 es 1550 BADIA_I_CUTCHET__maria 235445055|235445424
235444670 machen machen VERB VVINF _ 235444661 AUX ES 132423 834115 9700618 es 1550 BADIA_I_CUTCHET__maria 235445048|235445418
235444671 . . . $. _ 235444670 -PUNCT-ES 132423 834115 9700618 es 1550 BADIA_I_CUTCHET__maria _
```

Figure 3.1: German sentence in the CONLL-U-like format in the FEP6.

shallow tokenisation rules, which separated apostrophized prepositional articles in Italian and French by white space. This confuses other tokenisers, as they then categorize the apostrophe as punctuation and separate it from the prepositional article. Thus, Graën et al. (2014) classified the errors into different types and estimated their frequencies. This analysis resulted in the release of a revised version of the Europarl corpus, the *Corrected Structured Europarl Corpus (CoStEP)*[2]. The previously classified errors were widely corrected or removed. Additionally, texts only available in one language or translated by the speaker instead of a translator were dropped altogether. This smaller version of the *Europarl* corpus is tokenised and aligned on speaker turns. A chapter may contain just one or several speaker turns.

3.2 Data Format of the Corpus

Linguistic research requires more preprocessing, and therefore, several versions were derived from the *CoStEP*. The *Full European Parliament Corpus v. 6* (FEP6), used for the experiments in our thesis, contains a total of seven languages – German, English, French, Spanish, Italian, Finnish and Polish – and 136,298 aligned speaker turns (Graën, 2017, p. 9). However, Polish covers just 43,458 turns (Graën, 2017, p. 9). we worked with the corpus in an extended CONLL-U-like format[3] as shown in Figure 3.1 below. Each turn in the data set appears first in German, then follow English, Spanish, French, Italian and Finnish, before the next turn again starts with German. In the following, we will explain the columns containing information relevant to our work.

Column 1 holds a global ID for each token in the corpus. This is different from the original CONLL-U format, in which the 1^{st} column holds a global sentence ID. The ID simplifies cross-lingual token identification for the extraction of specific

[2]The *CoStEP* corpus is available at: `https://pub.cl.uzh.ch/wiki/public/costep/start`

[3]http://universaldependencies.org/docs/format.html (Accessed 5 April 2018).

examples, alignments and dependencies.

The data format further incorporates the word form for each token in the $2^{nd}column$. The individual tokens, as shown in column 2 of Figure 3.1, are the result of the tokeniser *Cutter 1.0* (Graën, 2017, p. 9). *Cutter 1.0* is equipped with rules specifically for the parliamentary domain and the languages available in the corpus. The language-specific rules are based on word forms, lemmas and POS tags. Thus, *Cutter 1.0* not only recognises word boundaries, but also sentence boundaries and sentence segment boundaries by colons and semicolons. After the tokenisation, a total of 240,637,915 tokens and 1,514,995 types can be identified (Graën, 2017, p. 9).

The corpus was tagged and lemmatised with the TreeTagger (TT) (Schmid, 1994) for all languages. Graën (2017, p. 9) applied the POS-tagging models available on the TT's webpage[4]. The lexicon of the tagger was extended by POS tags and lemmas for frequent words, particularly in German, in order to increase the accuracy and lemmatisation coverage of the tagger (Graën, 2017, p. 9). The lemmas are represented in column 3 and the TreeTagger POS tags in column 5 of the data format. It is noteworthy that the POS tag sets for the different languages differ greatly, as they depend on the material, with which the model was trained. The previously mentioned interlingual lemma disambiguation approach from Graën (2018), has not been applied to the data used in our experiments yet. Our data has been extracted from a pre-release version of version 6. As the lemmas are the core of our work and they are distributed together with the POS tags, we will closer investigate the tagging and lemmatisation process of the TT in section 3.2.1.1.

The 6^{th} column is empty and reserved for morphological information. We have enriched the data with a stand-off annotation, as explained in section 4.3.1.2.

The syntactic information is contained in column 7 and 8. (Graën, 2017, p. 10) reports that they trained the *MaltParser* (Nivre et al., 2006) for German and Italian, as there were no pre-trained models available, in order to derive the syntactical dependencies. For English, they applied the pre-trained parser model. The other languages were not parsed. For best results the *MaltOptimizer* (Nivre and Ballesteros, 2012) was applied for training the different parsing models. The parser returns the dependency relation for each token in column 8 to its dependency head in column 7.

Column 9 contains the country of origin of the turn speaker if applicable. Also given is the speaker's name in column 15 and the speaker's language in column 13. This information is not relevant for our project.

[4]http://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/

Column 10 contains the language-specific turn ID. This ID links all sentence segments across all languages to the speaker turn they belong to. Thus, all translations can be assigned to the specific turn via this ID, which enables turn-wise data extraction. Column 11, on the other hand, identifies a turn across all (translated) languages.

In order to retrieve entire sentences for a detailed data analysis, as for example in the gold standard, column 12 contains the sentence ID. The ID is unique for each sentence in the corpus.

Column 14, contains the ID of the language of the text. As opposed to column 13 the language does not refer to the originally-spoken language of the text, but to the language, in which it is presented in. When extracting data, we need to be able to identify and discern the different languages and sort the data accordingly.

The last column, 15, contains the pipe-separeted IDs of all aligned tokens of the token in question. Prior to word alignments, sentence segments were aligned with *hunalign* (Varga et al., 2005). Based on the sentence segment alignments, Graën (2017, p. 10) performed word alignment with GIZA++ (Och and Ney, 2003) and the Berkeley Aligner (Liang et al., 2006). Graën (2017, p.10) added the output of both aligners into the corpus. For the word alignment task, they mapped the different tag sets from the TT POS-tagging models to a universal tag set, represented in column 4. The universal tag set was designed by Petrov et al. (2012) and contains 12 POS tags. From the universal tags, they chose nouns, verbs and adjectives and adverbs as content words (Graën, 2017, p. 10). Word alignment was only performed on the lemmas of content words. Consequently, the closed word classes lack word alignment in FEP6, but the majority of ambiguous lemmas belong to an open class.

3.2.1 Lemmas in FEP6

3.2.1.1 Lemmatisation with the TreeTagger

As mentioned previously, the FEP6 was tagged and lemmatised with the TT and the POS-tagging models available its website[5]. In this section, we will explain how the TT POS-tagging models are built and how the lemmas are added.

The TT's POS-tagging models consist of decision trees obtained from annotated training data (Schmid, 1994). The decision trees are built on trigrams, meaning that the two preceding tokens are considered in order to determine the tag probabilities for the token in question, as represented in Figure 3.2.

[5] `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/` (Accessed 18 January 2018).

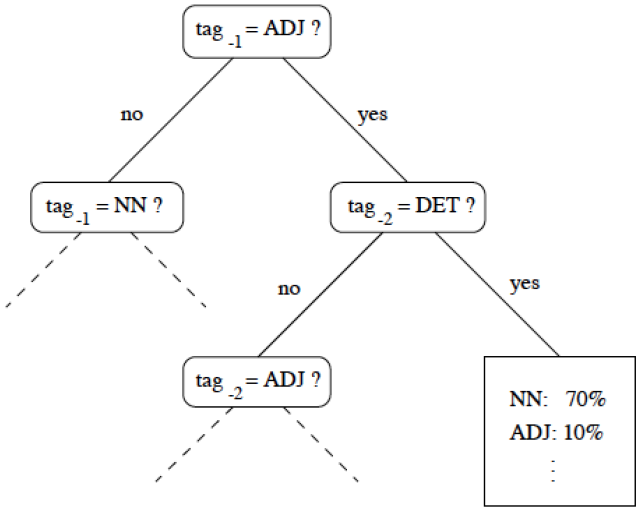

Figure 3.2: Illustration of a TT decision tree model as presented in Schmid (1994, p. 3)

Additionally, the tagger requires a lexicon that contains a priori tag probabilities for each word. The lexicon is split into a full form lexicon, a suffix lexicon and a default entry (Schmid, 1994). The full form lexicon has to be created from a tagged training corpus and contains the a priori tag probabilities for each word in the training corpus. If the model cannot find the word form as it is, or lowercased in the full form lexicon, it starts to search the character n-gram suffix lexicon. The suffix lexicon is also built on the training corpus and contains tag probabilities for suffixes. They are represented as tree structures, with the tag probability at the top node and the respective last character directly attached to the root. The model searches the tree from the root node along the branches to the leaf node, which directly attaches to a tag probability as in Figure 3.3.

If the model also fails to find the token ending in the suffix lexicon, the default entry is returned. The default entry is calculated by subtracting the tag frequencies of all leaves of the suffix tree from the tag frequencies at the root node. This explains how the TT learns the POS tag probabilities for POS tag sequences. On the other hand, the lemmas, are not learned by the POS tag model, but rather observed. The training corpus can, but does not have to contain lemmas alongside the word forms and the tags. If lemmas are provided, the tagger saves a word form/POS tag/lemma tuple. Whenever the model observes a word form/POS tag/lemma tuple with two or more different lemmas, it stores all lemmas that match the tuple separated by a pipe. This may lead to a large number of ambiguities, depending on the complexity

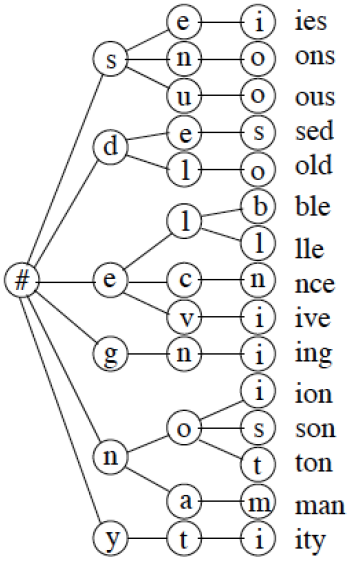

Figure 3.3: Illustration of a TreeTagger suffix tree model as presented in Schmid (1994, p. 5). The probabilities for the word classes for each suffix are not shown.

of the morphology of the language and the tag set. Hence, the TT does not really lemmatise the tokens, but rather provides lemma candidates that it has seen in the training corpus. Word form/POS tag/lemma tuples that do not occur in the training corpus are labelled as '⟨unknown⟩' by default or the word form can optionally be copied as lemma. Thus, there is no effort to lemmatise unseen word forms. The user can provide an external lexicon, which needs to contain a POS tag and a lemma for each token added, but the tagger never learns to lemmatise token/POS tag/lemma tuples that he had not seen in the training material. All in all, the TT is a very fast and useful tool for providing corpora with lemmas, as the lexicon can be extended and POS-tagging models for a variety of languages are provided on the website.

It is evident that the quality of the POS-tagging and lemmatisation in particular relies heavily on the annotation quality of the training corpus and that the TT is not built for lemmatising unseen words, but assigning POS tags to them. However, the information about training corpora and tag sets used for the different POS-tag language models is sparse. Table 3.1 provides an overview on what is known and unknown about the different models.

Apparently, there are no scientific publications about the training data of four out of 7 languages. This aggravates the identification of the error source in the POS tag model, but also the tagger lexicon that proposes the lemma candidates

Language	Resources (training corpus; tag set)
German	Training corpus: *unknown*; tag set: Stuttgart-Tübingen Tagset (STTS) (Schiller et al., 1995)
French	Training corpus: *unknown*; tag set designed by Achim Stein (Stein, 2003)
Italian	*unknown*, tag set designed by Achim Stein
Finnish	Training corpus: FinnTreeBank (Voutilainen et al., 2012); tag set: based on a simplified morphological tag set as reported by Graën (2018)
English	Training corpus: Penn Treebank Marcus et al. (1993); tag set: Penn Treebank tag set (Santorini, 1990)
Spanish	Training corpus: Spanish CRATER corpus (McEnery et al., 1997); tag set: EAGLES conformant tag set (McEnery et al., 1997)
Polish	Training corpus: National Corpus of Polish,(Przepiórkowski et al., 2008); tag set: unique morphological tag set

Table 3.1: Training corpora and tag sets, if applicable.

for seen word forms. It is possible that the lemma candidates were not manually annotated, but added by another morphological analyser or lemmatiser, which may have a considerable impact on the quality. This is the approach that was applied in the CRATER corpus, which was tagged with a newly trained POS-tagging model for Spanish of the Xerox tagger because McEnery et al. (1997) could not find a POS tagger for Spanish that met their needs. The Xerox tagger is based on a hidden Markov model and rules for ambiguity classes based on character n-gram suffixes (Cutting et al., 1992). As the Xerox tagger requires a lexicon for frequent word forms as well, the lemmas are a byproduct. Marcus et al. (1993) state nothing about their lemmatisation process for the English training corpus. However, as they had students to manually add the POS tags, it is very likely that they also manually lemmatised the word forms. The lemmas in the FinnTreeBank on the other hand are based on detailed instructions for manual POS-tagging and lemmatisation (Voutilainen et al., 2012). In summary, there is little known about the origin of the lemmas in the different tagger models.

3.2.1.2 Lemma Distribution of German, French Italian and Finnish

For further investigations and experiments we tried to assess the performance of the the TT with respect to lemmatisation. Furthermore, we calculated the number of ambiguous word forms. As the released version of the corpus no longer contains indications on unknown words to the model of the tagger, we reprocessed the data

language	German		French		Italian		Finnish		Spanish		English	
lemma candidates	tokens	types	tokens	types	tokens	types	tokens	types	tokens	types	tokens	types
0	763,214	179,396	3,039,516	61,053	143,532	72,118	454,549	128,747	787,829	105,759	526,642	49,256
1	37,100,969	188,067	40,455,135	96,066	37,463,101	123,765	27,336,454	519,074	40,978,866	126,722	39,425,664	132,498
2	220,687	487	225,488	330	548,405	803	627,609	45,725	78,890	299	31	3
≥ 3	336	8	6	1	9,113	6	34,546	3,170	167	3	0	0
total	38,085,206	367,958	43,720,145	157,450	39,458,151	196,692	28,453,158	696,716	41,887,594	232,783	39,952,337	132,498

Table 3.2: Frequency distribution of lemmas for the six languages considered in our experiments in FEP6. 'tokens' refers to tuple tokens and 'types' refers to tuple types. The column 'lemma candidates' contains the amount of lemma candidates with 0 for unseen tokens and ≥ 3 for more than 2 lemma candidates.

for German, French, Italian and Finnish. The number of unknown tokens in the released version is smaller, as we did not apply an extended tagger lexicon. Hence, we can asses the coverage of the TreeTagger models as downloaded from the website.

Table 3.2 contains the absolute number for every single occurrence of word form/POS tag/lemma tuples, reffered to as tuple tokens, and for the number of distinct word form/POS tag/lemma tuples, reffered to as types. As shown in Table 3.2, the TT cannot lemmatise 763.214 (2%) tokens out of total 38,085,206. This results in a token level coverage of 98%. 221,023 (0.5 %) tuple tokens are left ambiguous with two or more lemma candidates. The tuple type *Arbeit|Arbeiten* in example 3.1 occurs 1,971 times as ambiguous tuple token[6]. More than two lemma candidates can be found for 338 tuple tokens of 8 different tuple types, as in example 3.2.

(3.1) *Arbeiten* *Arbeit|Arbeiten* *1,971*
token lemma candidates count

(3.2) *Winden* *Wind|Winde|Winden* *6*
token lemma candidates count

This results in 436 different types of tuples with two lemma candidates and 8 different types of tuples with more than two lemma candidates, as shown in example 3.2. Tuple types subsume identical tokens with an identical POS tag and an identical lemma. Example 3.3 represents column 2 in Table 3.2 and illustrates that the token 'Mitteln' occurs 3,369 times with the ambiguous lemma *Mittel|Mitteln*.

(3.3) *Mitteln* *Mittel|Mitteln* *3,369*
token lemma candidates count

As Graën (2018, p. 31) notes, a large number of ambiguities result from nominalisations of verbs and inflected nouns — often dative plural — that match in

[6]The POS tags are not shown in the examples, but were considered in the counts.

their form. Examples are *Ringen* 'ring' (dative plural) / 'struggling' (derived noun) and *Morden* 'murder' (dative plural) / 'murdering' (derived noun). However, it may occur as well with entirely unrelated words, such as *Dosen* 'can' (all cases plural) / 'dose' (all cases plural). Other cases such as *Leinen* 'leash' (all cases plural) / 'linen' (singular) / 'Leinen' (surname of person) do have three possible meanings, but only two lemmas 'leash' and 'linen' differ in their form. The difference between 'Leinen' (surname of person) and 'linen' (singular) is a word sense issue and not a lemmatisation issue.

As shown in Table 3.2 in column 4, the TT cannot lemmatise 3,039,516 (6.9 %) of 43,720,145 French tuple tokens in total. This results in a tuple token level coverage of 93.1 %. Another 225,488 tuple tokens have 2 lemma candidates (0.5 %) and only 6 tuple tokens feature more than 2 lemma candidates.

When inspecting the gold standard, a large number of ambiguities in French appear to co-occur with verbs. French is an inflected language and French verbs are divided into different conjugations. The '-ir' conjugation can be extended with '-is' or '-iss' and these extensions may lead to ambiguities with the '-er' conjugation in the indicative plural, such as in example 3.4, in which *pressent* is assigned with the candidates *pressentir*|*presser* ('to anticipate' and 'to flock').

(3.4) *Nous devons décupler nos efforts pour envoyer nourriture,*
we must reinforce our efforts to send food
médicaments, médecins et apporter notre soutien aux milliers
drugs doctors and give our support to thousands
de réfugiés qui se pressent en Tunisie et en Égypte.
of refugees which flock to Tunisia and Egypt
'We need to reinforce our efforts to send food, drugs and doctors give support to the thousands of refugees, which flock to Tunisia and Egypt.'

The same accounts to verbs from the '-er' and the irregular conjugation, such as in example 3.5, in the third person plural indicative and the present participle.

(3.5) *moulant moudre*|*mouler*
token lemma candidates
Les moulins de la procédure de recours moulent lentement .

Of course there are also ambiguities due to nominalisations, such as *frais* 'fresh air' (nominalised adjective masculine singular) / 'spawn' (masculine plural noun). Additionally, the French definite article *le/la* is ambiguous when clitisized before a vowel, as *'l*, or always in the plural *les*.

The TT has a token-level coverage of 96.4 % for Italian, as 1,437,532 out of 39,458,151 tuple tokens are unknown to the Italian tagger model.

```
word form c lemma candidates c1 c2
Etiketten 117 Etikett|Etikette 256 6
Fächer 22 Fach|Fächer 130 0
Fächern 14 Fach|Fächer 130 0
Fällen 4432 Fall|Fällen 21740 0
Falten 1 Falte|Falten 0 0
```

Figure 3.4: The first column contains the word forms and the second column contains the number of times the word form with the ambiguous lemma in column 3 occurs. Column 4 and 5 contain the cooccurrence counts for the unambiguous lemma candidates.

3.3 Gold Standard and Lemmatisation Quality Measurements

For the different experiments, a gold standard in four languages — German, French, Italian and Finnish — was established in order to have manually annotated data to test the performance of the machine learning approach and to measure the quality of the TT lemmatisation[7]. The gold standard consists of two, manually annotated parts. Part A was extracted for the evaluation of the distant semi-supervised approach. Part B is limited to 100 observations for all languages and was established for the evaluation of the active learning approach[8]. Hence, we extracted all lemmas with evidence for both lemma candidates as unique candidate in the corpus for part A. The word form *Etiketten* in Figure 3.4 occurs 117 times with the lemma candidates *Etikett* and *Etikette* in the corpus, while 256 other word forms have *Etikett* as sole lemma candidate and 6 other word forms have *Etikette* as sole lemma candidate.

We then applied a simple Naive Bayes classifier from the NLTK-toolkit by Loper and Bird (2002) on the extracted data, in order to have a pre-annotated lemma for the annotators so that they can efficiently accept or correct. The annotated lemmas were randomly subsampled with a maximum of 10 examples per ambiguous lemma. If the corpus contains less than 10 examples, all of them were included. This cap of maximally 10 instances per lemma was meant to keep the annotation effort within a limit. The annotators were presented with the word form in question, the ambiguous lemma, the proposition of the Naive Bayes classifier and the word

[7]By now the SPARCLING project has ended and version 9 of the corpus is released. The gold standard was also derived from version 6, which differs with respect to token IDs, lemmatisation and sentence alignment. Hence, adjustifications would be necessary to map the gold standard to version 9.

[8]In Chapter 5 we will explain why part B of the gold standard can be extended.

WF	LEMMA	DISAMB	R	SENT 1	WF	SENT 2
abgewogen	abwiegen|abwägen	abwägen	1	Des Weiteren wurde ein Änderungsantrag über ein System mildernder Umstände, bei dem die Faktoren gegeneinander	abgewogen	werden, verworfen.

Table 3.3: An example from the input format of the gold standard creation of part A. The column abbreviations will be used for the following examples alike: WF = word form, DISAMB = disambiguation, R = rating, SENT 1 = first part of the sentence, SENT 2 = second part of the sentence.

WF	LEMMA	R	SENT 1	WF	SENT 2
tecniche	tecnica|tecnico	tecnica	potremo farlo promuovendo misure volte a garantire la sicurezza alimentare nonché ammodernando le	tecniche	di produzione .

Table 3.4: An example from part B of the gold standard, as presented to the annotators.

form in its sentence context. The column labelled as "CORRECT" was left blank for the annotator to efficiently evaluate the Naive Bayes' output. A rating of "1" means that the classifier has delivered the correct answer as in example 3.3, a rating of "0" declares the classifier's answer to be wrong. We initially planned to evaluate the Naive Bayes on German only and did not have more complex languages, such as Finnish with frequently more than 2 lemma candidates, in mind or that the tagger would not cover all possible lemma candidates. Thus, we also accepted manually entered lemma candidates in column 4.

Part B of the gold standard also contains word form/POS tag/lemma tuples, of which not all lemma candidates occur as unique lemma candidate in the corpus. We realised that active learning could be an opportunity to increase the coverage of our semi-supervised machine learning approach, and hence, we need data for the development and evaluation of the extended approach. For this application, we extracted all the tuples that occur at least 15 times in the corpus, but only some or none lemma candidates occur as unique lemma candidates. Again a maximum of 10 examples per tuple were picked randomly and from the remaining examples, just 100 were picked randomly for manual evaluation, as the active learning approach, which we will explain in Chapter 5, can automatically extend the gold standard with each iteration. The volunteers could not accept or reject a recommended lemma in this partition. They were simply presented with the lemma candidates and had to fill in the correct answer. This made the task more tedious, as shown in Table 3.4.

The German and French gold standards were established by ourselves, while Finnish and Italian were established by three different annotators[9]. In order to

[9]The annotators for Italian are Italian native speakers with academic degree in linguistics. The

	1	**0**	**added lemma**	**Total**
1	24	0	0	**24**
0	2	2	0	**4**
own lemma	0	0	7	**7**
Total	**26**	**2**	**7**	**35**

Table 3.5: Comparison between annotator 1 and annotator 2 of Italian across 35 lemmas.

make sure that the idea behind the task is well understood and executed alike by all annotators, we created a guideline with instructions and examples. These guidelines are disclosed in Appendix C.

We compared the results and the performance of the Italian annotators by calculating an unweighted Cohen's Kappa value as shown in 3.6, by letting them independently annotate 35 examples.

(3.6)

$$k = \frac{p_0 - p_c}{1 - p_c}$$

Cohen (1960) suggests to test such categorial data, as in the gold standard, for interrator agreement. The resulting Kappa value measure the inter-annotator agreement.. Additionally, a rather negative outcome could also hint towards insufficiently written annotation guidelines. The resulting values may range from -1 to 1. According to Cohen (1960), results below 0 are improbable in practice and a value of 0.01-20 may be interpreted as none to slight agreement, 0.21-0.40 as fair, 0.41-0.60 as moderate, 0.61-0.80 as substantial and values above 0.80 as almost perfect agreement. Thus, results above 0.60 shall be aimed for, as then a majority of the data in the small subsample should be annotated correctly. Table 3.3 contains the 35 lemmas in common, which were compared in order to calculate the Cohen's Kappa score.

The Cohen's Kappa score of the above comparison is 0.87 with a 95 % confidence interval between 0.7 - 1.0. Therefore, we can assume that they worked through the samples carefully. In case of disagreement within the 35 mutual lemmas, both annotators were asked to discuss their judgements and agree on a variant. In both cases they came to an agreement. We would not expect deviations of the result on

annotator for Finnish is a Swiss German native speaker and Finnish L2-speaker with an academic degree in linguistics.

a larger comparative set because it usually takes little effort to select the correct lemma candidate of content words for people with a linguistic background.

Generally, the gold standard also contains lemma candidates, which are not proposed by the TT, as the annotators were allowed to add additional lemma candidates, whenever the correct output was not in the tagger output. Depending on the lexicon of the tagger, some ambiguities might have been missed or wrong POS tags lead to incorrect lemma candidates. Both cases are relevant to estimate the lemmatisation quality of the TT. The quality will be evaluated by precision (see 3.7), recall (see 3.8) and the F_1-measure (see 3.9), which is the harmonic mean of precision and recall.

(3.7)

$$precision = \frac{TP}{TP + FP}$$

(3.8)

$$recall = \frac{TP}{TP + FN}$$

(3.9)

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

3.3.1 German Gold Standard and Lemmatisation Quality

For the German gold standard we sampled a total of 809 lemma occurrences with evidence for all lemma candidates for part A and 100 lemma occurrences with evidence for not all lemma candidates for part B. This makes a total of 909 manually annotated instances. One lemma candidate was missing, due to a spelling error in the text. In the sentence context given in example 3.10 it becomes evident that *Bürgen* ('baling', noun) has to be a mistake and it should be *Bürger* ('citizen', noun) instead. Additionally, the tagger suggests the lemmas *ausstatten* ('to equip') and *statten*, a lemma that does not exist in the German lexicon. Nevertheless, the TT assigns the lemma *statten* also as sole, independent lemma candidate.

	Detected Lemma Candidates	Undetected Lemma Candidates
Incorrect Lemma Candidates	FP = 22	TN = 0
Correct Lemma Candidates	TP = 1,785	FN = 11

Table 3.6: Comparison between correct lemma candidates and detected lemma candidates by the German TT model.

(3.10) *Bürger werden drittens mit dem Bericht die Regierungen dazu*
citizens will thirdly with the report the governments to
aufgerufen, das Recht auf Internetzugang auch den ärmsten
calls the right to internet access also the poorest
Bürgen[sic] in den abgelegensten Regionen einzuräumen;
baling[sic] in the most remote regions give
'Thirdly, the report calls on governments to give the right to access the Internet also to the poorest baling[sic] in the most remote regions;'

It appears to be an erroneous entry in the tagger lexicon, which occurs 50 times as unique lemma candidate, as opposed to 3,126 for *gestatten* in FEP6. The results of this evaluation are presented in 3.6.

The manual annotation of the gold standard results in a precision of 98.8 % and a recall of 99.4 % with an F_1-score of 99.1 % for the TT lemmatisation.

3.3.2 French Gold Standard

The French data sample was much larger at the beginning, but manual scrutiny revealed that 159 out of 403 tokens only have seemingly ambiguous lemmas. The word form *payer* ('to pay') for example, obtained the lemma *payer*|*payer*, which is obviously not a real ambiguity and must be due to problems in the creation of the TT model. We excluded these cases from the gold standard. There is a certain risk that some of these word forms are nonetheless ambiguous, but the expenses of manually correcting the lemmas were estimated higher. This affects a total of 9,860 word form/POS tag/lemma tuple tokens or 28 tuple types and 225,494 ambiguous tuple occurrences and 336 ambiguous tuple types remained. These pseudo-ambiguous lemmas can be corrected with simple search and replace methods.

The remainder of the French gold standard contains 244 lemma occurrences in part A and 100 occurences in the part B. This results in a total of 344 lemma occurrences. In 24 occurrences, a lemma candidate had to be added manually. For some cases, the tagger did not propose the correct candidate, as it has assigned the wrong POS tag as in example 3.11. *cru* ('raw') is clearly an adjective, but the proposed lemma candidates *croire* ('believe') and *croître* ('grow') are verbs, as

they have the same past participle *cru*. If the tags were correct, the verb lemma candidates would not be an option.

(3.11) *Cette catégorie couvrirait les fruits et les légumes crus, certains produits laitiers tels que*
This category cover.FUTURE the fruits and the vegetables raw, some produce dairy such as
le yaourt, et quelques bières.
the yogurt and some beers

'This category will cover raw vegetables and fruits, some dairy produce, such as yogurt, and some beers.'

This happens even more often with unrecognised named entities, as in 3.12. *Argentine* ('Argentina', noun) clearly refers to the country in this context, and is not an adjective. The resulting lemma candidates are *argentin*|*argentine* ('Argentinian', masculine and 'Argentinian', feminine).This is also interesting because the masculine and feminine form of an adjective are usually not regarded as separate lemmas, unless the female form carries an (additional) different meaning.

(3.12) *En outre, le soutien aux mesures gouvernementales anti-populaires en*
additionally the support for measures governmental anti-popular in Argentina
Argentine et la peur que celle -ci échappe à l' emprise du FMI apparaissent
and the fear that they escape of the influence of FMI appear clear
clairement.

'Additionally, the issue of the support for anti-populistic government measures in Argentina and the fear that the FMI loses its influence becomes evident.'

There is a considerable amount of lemma candidates also with noun tags that we would not consider to be independent lemmas, such as *clémentine*|*clémentines* ('clementine' and 'clementines'). Both lemma candidates refer to exactly the same concept and thus it is unnecessary to have the inflected plural form in the lemma inventory. Such inflected lemma candidates are most likely a lemmatisation error in the training corpus. Even more so when the word is inexistent in the French lexicon, such as *liser*, which is proposed as lemma candidate alongside *lire* ('read') for *lisent* ('read', 3rd person plural present).

Another error source are typos, such as in example 3.13. *plut* indeed can be either past participle of *plaire* ('like') or *pleuvoir* ('rain'), but the context reveals that it has to be *plus* ('more') instead of *plut*.

(3.13) *Comme vous l' avez montré, Monsieur de Maizière, notre réunification (la réunification de l'*
as you it have shown Mister de Maizière our reunion the reunion of the
Allemagne) a débuté encore plut[sic!] tôt, puisque nous avions modifié notre attitude et
Germany has started again more early because we have.PAST changed our attitude and
notre approche de notre histoire européenne commune et de notre avenir européen commun.
our approach from our history European united and of our future European united

'As you, Mister de Maizière, have shown, our reunion (the reunion of Germany) has already started earlier because we had modified our attitude and our approach towards our united European history and towards our united European future.'

The results of this evaluation are summarised in Table 3.7. Consequently, the precision of the French lemmatisation is 89.24 % at a recall of 95.46 %, resulting in an F_1-score of 92.0 %.

	Predicted Positive	Predicted Negative
Negative Cases	FP = 71	TN = 0
Positive Cases	TP = 589	FN = 28

Table 3.7: Comparison between correct lemma candidates and proposed lemma candidates by the French TT model.

3.3.3 Italian Gold Standard

Part A of the Italian gold standard sample contains 1,052 and part B 100 ambiguous lemmas. The first annotator was given 562 lemmas and the second annotator 525 of part A, of which 35 overlapped. Both annotators also disambiguated 50 instances from part B of the gold standard. This results in a full gold standard containing a total of 1152 lemma occurrences.

A total of 79 lemma candidates, which can be subsumed into 21 lemma candidate types, had to be complemented manually by the annotators. This included word forms of rather frequent verbs such as *portare* ('to carry') in example 3.14. The context reveals that it is incorrectly tagged as noun, and thus the tagger proposes the candidates *porto* ('harbour', noun masculine) and *porta* ('door' noun, feminine), instead of the verb *portare* ('carry').

(3.14) *La situazione nei paesi a cui la relazione si riferisce (parlo principalmente della zona*
the situation of the countries to which the report itself refers talking mainly of the region
orientale) è molto dinamica e porta, prevedibilmente, molte esperienze nuove.
eastern is very dynamic and carries predictably many experiences new
'The situation of the countries to which the report refers to (mainly talking about the Eastern zone) is very dynamic and promises, predictably, many new experiences.'

This also happened several times with unrecognised named entities, as they are often tagged as regular nouns, if present in the tagger lexicon. So is *Rosa* a frequent German and English personal name, while *rosa* in Italian may be either a personal name (capitalised) or refer to the flower 'rose'. In the following example 3.15 it clearly refers to a person.

(3.15) *Riteniamo che delle elezioni trasparenti, democratiche e imparziali - la libertà dei*
we believe that such elections transparent, democratic and impartial the freedom of the dissenting
dissenzienti per citare Rosa Luxemburg - siano un requisito fondamentale
to quote Rosa Luxemburg - are a requirement fundamental for
per stabilire rapporti con la Bielorussia e tutti gli altri Stati.
relationships establish with the Belarus and all of the other states
'We believe that transparent, democratic and impartial elections - the freedom of dissenting people to quote Rosa Luxemburg - are a fundamental requirement in order to establish relationships with Belarus and all other states.'

Example 3.15 is also interesting because the other proposed lemma candidate is *roso*, which is an inexistent word in Italian. *roso* is one of numerous errors in the

	Predicted Positive	Predicted Negative
Negative Cases	FP = 187	TN = 0
Positive Cases	TP = 2045	FN = 72

Table 3.8: Comparison between correct lemma candidates and proposed lemma candidates by the Italian TreeTagger model.

TreeTagger lexicon for Italian. Moreover, some ambiguous lemmas are affected by typos. *coni* (either 'embossing' or 'cone') in example 3.16 should be spelled *con i* ('with their'), as revealed by the context.

(3.16) *Capirete che questi elogi vanno anche a tutti coloro che hanno dato il*
you will understand that this praise goes also to all those who have given their contribution
loro contributo, ad esempio i deputati europei che si sono
for example the assemblymen European who themselves interest.Past into the achievement
interessati all' operato del Mediatore europeo e che mi hanno incoraggiato
of the mediator European and who me have encourage.PAST cone[sic] their advice
coni[sic], loro consigli e i loro pareri.
and their opinions

'You will understand that this praise goes also to all of those who have given their contribution, for example the Member of the European Parliament, who have been interested in the work of the European mediator and who have encouraged me with their advice and their opinions.'

The errors are summarised in Table 3.8. The Italian TT model has a precision of 91.62 % and a recall of 96.6 %. This results in an F_1-score of 94.0 %.

3.3.4 Finnish Gold Standard

The Finnish sample for part A originally included more than 1000 tokens. However, as only one annotator could be found, only the first half was manually disambiguated. The resulting part A of the gold standard contains 664 disambiguated tokens. Part B consists of 100 lemmas, which results in a total of 764 lemmas. A total of 32 word form lemma-tokens, or 17 word form lemma-types, had to be added manually. As Finnish is a highly agglutinative language, it is particularly prone to building homonymous word forms. Thus, the tagger had considerable trouble with named entities from foreign languages, such as the French *Rue de la Loin* as shown in example 3.17.

(3.17) *Milloin julkaistaan Rue de la Loin arkkitehtikilpailun*
when publish.FUTURE Rue de la Loin architectural competition
tulokset?
results

'When will the results of the Rue de la Loin architectural contest be published?'

The annotator was asked whether she discovered any lemmas that could be

	Predicted Positive	Predicted Negative
Negative Cases	FP = 0	TN = 0
Positive Cases	TP = 732	FN = 32

Table 3.9: Comparison between correct lemma candidates and proposed lemma candidates by the Finnish TT model.

lemma candidates	TT	Gertwol
0	179,396	54,371
1	188,067	341,702
2	489	5,808
≥ 3	8	48
total	367,960	401,929

Table 3.10: Comparison of lemma frequency distributions between TT and GerTwol for German.

lemmatised wrongly due to tokenisation or spelling mistakes. According to her, none the like can be found. Table 3.9 shows the errors of the TT. The Finnish TT mode has an estimated precision of 100 % and a recall of 95.8 %, which results in an F_1-score of 97.8 %

3.4 Lemma Coverage in German

We wanted to investigate whether a morphological analyser may discover yet unknown lemma candidates. For this purpose, we applied GerTwol, the previously mentioned morphological analyser for German, on the German part of our corpus. However, as the original POS tags of GerTwol differ drastically from the STTS tag set, which is applied by the TT, we used the GerTwol to STTS-tag mapping pipeline by Clematide. Additionally, we applied Volk (1999)'s rules for choosing the right lemma. He forged a set of rules that score the different segmentation boundaries returned by GerTwol and return the lemma candidates with the lowest penalty score. The remaining word form analyses were grouped into word form/POS tag/lemma tuples similar to the TT, as the TT can only assign a lemma when the word form and the assigned tag have a lemma in the lexicon, and GerTwol sometimes outputs several POS categories for one word form, as for example *Spielen* can be either *spielen* ('play') or *Spielen*. This yields 3 times fewer unknown, but also twice as many known tokens and almost 10 times more ambiguous word form/POS tag/lemma tuples, as shown in Table 3.10.

This overall comparison does not tell much yet, apart from the fact that GerT-

wol reduces the amount of unknown tokens, but also increases the amount of ambiguities. To further examine this, we compared the TT output and the GerTwol output by word class by accepting only word form, POS tag cooccurrences from GerTwol that also occur in the TT lexicon, as we trust the TreeTagger's ability to assign correct POS tags in context. We chose nouns, including proper nouns, adjectives and verbs, all inflected and uninflected forms because they are highly marked for inflection and open classes with new coinages and compounds. Table 3.12 shows that GerTwol increases the coverage the most with nouns, as it outputs three times less unknown lemma candidates and almost twice the amount of unambiguous lemma candidates as compared to the TT. The amount of ambiguous nouns is almost 10 times higher. For the adjectives, GerTwol produces slightly more than half of the amount of unknown lemma candidates, but only a quarter more unambiguous candidates, while the amount of ambiguous lemmas increases by the factor 10.The numbers for the verbs are almost identical for the TT and GerTwol.

As the output of the TT and GerTwol differs substantially, we decided to manually evaluate 100 instances per word class, for which GerTwol finds more lemma candidates than the TT has in its lexicon[10]. However, we only found 54 verbs with additional lemma candidates. As Table 3.11 reveals, 20 % of the new nominal lemma candidates and 30 % of the verbal lemma candidates are valid options, while no additional adjectival candidates could be found. It turns out that this is due to a mapping problem between GerTwol and the STTS tag set. While GerTwol assigns the category "present participle" to deverbal adjectives, such as *weinend* ('crying'), which is wrongly mapped to the predicative adjective class "ADJD" from the STTS tag set because there is not present participle. GerTwol suggests, the positiv adjectival form *weinend* as well as the verb infinitive *weinen* ('to cry'). This leads to a large number of false positives. Additionally, GerTwol mostly suggests the attributive, as well as the entirely unmarked predicative form, such as *still* ('silent', predicative) and *stille* ('silent', attributive), while the TreeTagger prefers the entirely unmarked form in all evaluated cases. Additionally, GerTwol builds the positive form of compound adjectives with a graded head, such as *nächstfolgend* ('next', superlative adjectival head) and *nahfolgend*[sic] (positive adjectival head), which is technically possible, but semantically wrong. A similar phenomenon can be observed with nominal compounds, where we have suggestions, such as *Spezialfirma* ('specialist company') and *Spezialfirmen*[sic] for *Spezialfirmen*, as *firma* ('company') is a noun and *firmen* can be either the plural of the former or a a verb (´to confirm'), which can be nominalised and compounded, but does not make any sense. The verbs tend to be problematic, when they build past participle with the prefix *ge-*, which

[10]We only checked instances for which the TT suggests at least one lemma candidate.

	nouns	verbs	adjectives
absolute	22	16	0
percentage	22 %	30 %	0

Table 3.11: Amount of true new ambiguities from GerTwol.

word class	nouns		adjectives		verbs	
lemma candidates	Gertwol	TT	Gertwol	TT	Gertwol	TT
0	49,475	141,324	12,523	22,862	7,095	7,243
1	184,801	96,916	55,157	45,412	27,119	26,964
2	4,346	382	596	2	102	109
total	238,623	238,623	68,276	68,276	34,316	34,316

Table 3.12: Comparison of lemma frequency distribution across word classes between GerTwol and TT.

can also function as derivational prefix, such as *gemahnt* ('admonished', past participle), which is analysed as *mahnen* ('to admonish') and *gemahnen*[sic]. Additionally, GerTwol's lexicon also contains some very archaic verbs, such as *schwären* ('to fester'), which are highly unlikely to appear in most modern German texts. Generally, GerTwol still tends to overanalyse, although we applied some filtering to the output. The remaining valid lemma candidates show that particularly with nouns, but also verbs, the coverage of the TT lexicon could be better with respect to additional ambiguities, while it appears to be sufficient for adjectives. GerTwol is an invaluable source for the lemmatisation of word forms, which are unknown to the TT, and that the TT is a frequently used standard tool for tagging and lemmatisaion of German texts for good reason.

4 Distant Semi-Supervised Machine Learning Approach

In this chapter we present our semi-supervised machine learning approach to lemma disambiguation. We firstly illustrate the underlying idea, as well as the potential drawbacks. We also discuss the feature extraction process, the feature selection and, the selection of the machine learning algorithms and their evaluation.

4.1 Concept of the Distant Semi-supervised Machine Learning Approach

The training of our models with machine learning algorithms is based on the assumption that the individual lemma candidates of the ambiguous lemmas, also occur as unambiguous lemmas in the corpus. This underlying idea is very similar to the one in the previously mentioned system for WSD by Lefever and Hoste (2014). This is true for some, but not all ambiguous lemmas as mentioned previously. In German, lemmas such as *Arm*/*Arme*, both lemma candidates *Arm* ('arm') and *Arme* ('pauper') occur independently in the corpus. However, for lemmas, such as *Akt*/*Akte* only *Akt* ('act') can be found unambiguously, and for examples such as *Ahne*/*Ahnen* ('ancestor' and 'anticipating') none of the two. Still, we decided for a distant semi-supervised machine learning approach, in which all sentences containing an independent lemma candidate of an ambiguous lemma are used for the training of a model, as this approach does not need any manually annotated data. In this distant semi-supervised approach, we assume that the unambiguous lemma candidates are assigned correctly and exploit them as labels for training. This is essentially a classification task. We have a set of known labels and train an algorithm to classify unseen data correctly.

The machine learning algorithm will never observe an ambiguous lemma in the training set. This could pose a problem, as the environment of the ambiguous lemma and its individual candidates could be different. This could happen, for example, when one of the parts has a different meaning depending on its grammatical gender

	German	French	Italian	Finnish
complete evidence	139 (28 %)	219 (66 %)	306 (39 %)	1,349 (3 %)
incomplete evidence	356	112	474	47,546
total	495	331	780	48,895

Table 4.1: Distribution of evidence for lemma candidates of ambiguous word form/POS tag/lemma tuple per language.

or is a homonym. An example for the former from German is the ambiguous word form *Erben* 'to inherit' (nominalised verb, nominative singular neuter) / 'heritage' (nominative plural neuter) and 'heir' (nominative plural masculine). Although *heir* and *heritage* are semantically related, they are not identical. However, we assume that these cases are rare and follow the"one sense per discourse" hypothesis, alleged by Gale et al. (1992) and supported by Yarowski (1995). The latter found evidence that words express the same sense with a probability above 99 %, if occurring in the same discourse and collocation[1].

Table 4.1 shows the distribution of unambiguous lemmatisations of ambiguous lemmas. There is a considerable amount of evidence for the German (28 %), French (66 %) and Italian (39 %) ambiguous word form/POS tag/lemma tuples, in the corpus. However, only a very small proportion (3 %) of ambiguous Finnish word form, POS tag, has sole independent lemma evidence for all lemma candidates. This means that the distant semi-supervised may not be the best choice for Finnish, as the coverage will be very low. We further investigated whether we could reduce the multiple ambiguous lemmas to the lemma candidates with evidence in the corpus by omitting the candidates without evidence in the corpus. However, we found that 85 % of lemmas have no evidence for any lemma candidate in the corpus and only 12 % have evidence for at least one candidate. This means that we had to tackle at least 85 % of the ambiguous word form/POS tag/lemma tuples with the active learning approach (introduced in Chapter 5)[2]. For German, Italian, and French, we have a fair chance of lemmatising a considerable amount of lemmas with a fast and economical method that does not require manual annotations.

We do not use a single machine learning algorithm, but experiment with a multitude of machine learning algorithms. Firstly, we want to fulfil the disambiguation task for four different languages. They are as different as pears and stones and, thus, require an adjusted treatment. This means that we have to develop multiple feature sets and train the models individually. Secondly, we treat each ambiguous

[1]Collocation is meant in the traditional way: two words cooccurring in the same location.

[2]In Chapter 5 we will also discuss, why the active learning approach is not very appropriate.

lemma as a separate classification problem. This leads to a pipeline as depicted in Figure 4.1. We first generate and extract the features for all languages, transform them into feature vectors for each ambiguity problem, train a model and then apply the predictive model on the ambiguous lemmas.

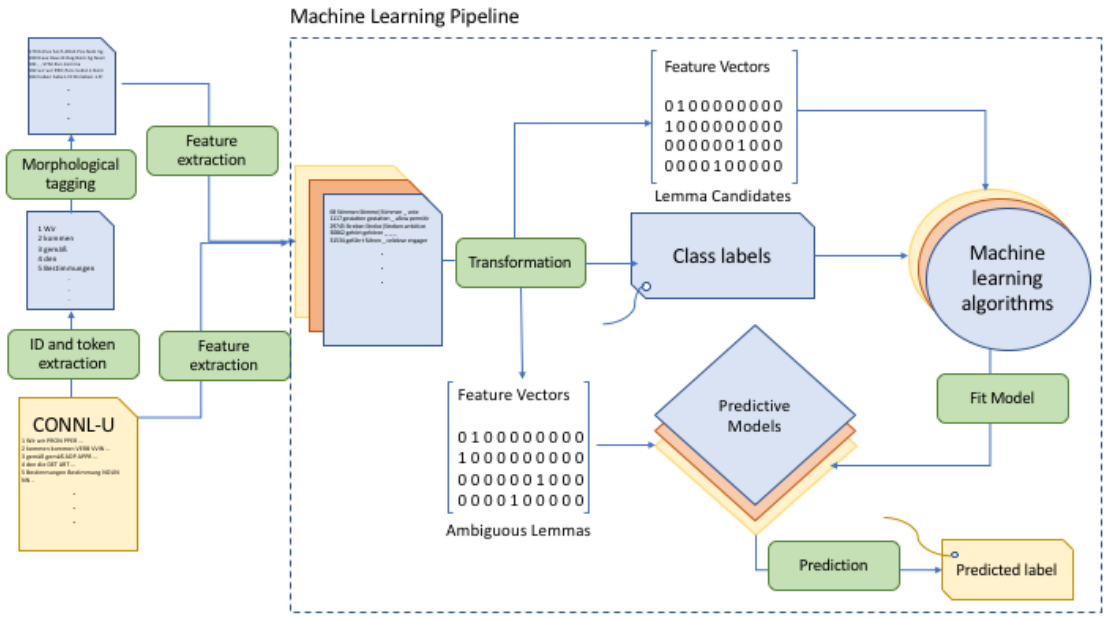

Figure 4.1: Graphic representation of the pipeline from the corpus towards the prediction of the disambiguated lemmas.

Most of the pipeline is implemented with the *scikit-learn* Python library by Pedregosa et al. (2011), which holds a number of different machine learning algorithms, as well as feature transformation tools.

4.2 Machine Learning Algorithms

We implemented and tested several machine learning algorithms, as the choice of algorithm has impact on the performance of the resulting models.

4.2.1 Naive Bayes

Naive Bayes is a rather simple, probabilistic classifier based on the Bayes theorem. A Naive Bayes classifier tries to determine the probability of a hypothesis H to be true given the context E, as shown in 4.1.

(4.1)

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

We can multiply the probability of E given H by the probability of H and divide it by the probability of E, as it assumes all contexts E to be independent from one another (Witten et al., 2011, p. 92). Regardless of, or maybe for its simplicity and naive assumption of an independence of features given the class, the algorithm performs comparably well with other, more complex algorithms (Navigli, 2009, p. 18).

We chose the `MultinomialNB()` classifier from the `sklearn.naive_bayes`[3] class, as it accepts more than two classes. Therefore the Naive Bayes formula is slightly adapted[4], as shown in equation 4.2. As the `MultinomialNB()` classifier cannot handle negative scalar data, we have switched to the `BernoulliNB()`[5], as it can also handle negative scalar data, such as in our word embeddings.

(4.2)

$$\hat{\theta}_{yi} = \frac{\sum_{x \in T} x_i + \alpha}{\sum_{i=1}^{|T|} N_{yi} + \alpha n}$$

The new equation estimates the parameter θ_{yi} by a smoothed version of the maximum likelihood, in which the number of times a feature i is observed with class y in the training set T multiplied with the smoothing factor α is divided by the total number of features for class y multiplied by the smoothing factor α and the total number of training examples n. The smoothing factor with $\alpha > 1$ accounts for features that have never been seen during training and prevents probabilities of zero in the test set. We implemented the standard smoothing, which is a Laplace smoothing[6] with $\alpha = 1$.

[3] `http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html#sklearn.naive_bayes.MultinomialNB` (Accessed 13 April 2018).

[4] `http://scikit-learn.org/stable/modules/naive_bayes.html#gaussian-naive-bayes`, (accessed 13 April 2018).

[5] `http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html` (Accessed 13 April 2018).

[6] Also called additive smoothing. In this case, we just add one to every feature count to avoid zero probabilities in unseen data of the test set.

4.2.2 Support Vector Machines (SVM)

SVMs have been reported frequently to yield good results in WSD and other text classification tasks (Navigli, 2009; Raschka, 2015). The algorithm behind SVM tries to find the maximum margin hyperplane (Witten et al., 2011, p. 224). The maximum margin hyperplane is a model, which linearly separates two classes correctly (Witten et al., 2011, p. 224). The instances of both classes that are closest to the maximum margin hyperplane are the support vectors, and each class has at least one support vector, but usually more. The support vectors make all other data vectors irrelevant, as they uniquely define the margin (Witten et al., 2011, p. 224). The smaller the margin, the more prone is a model to overfit, as it tends to overgeneralize (Raschka, 2015, p. 70).

According to Witten et al. (2011, p. 225), the maximum margin hyperplane can be defined as in 4.3.

(4.3)

$$x = b + \sum_{\text{i is support vector}} \alpha_i y_i a(i) \bullet a$$

This function computes the maximum margin hyperplane x by multiplying the dot product of a test vector a and a support vector *a(i)* by its class value y_i and the parameters b and α_i. The parameters b and α_i define the hyperplane and have to be learned alongside the support vectors by the algorithm (Witten et al., 2011, p. 225). It is possible to model nonlinear class boundaries by increasing the dimensionality of the vector space. However, we will not go into detail with nonlinear models, as we have only worked with the linear SVM algorithm.

We implemented the linear SVM algorithm with the `LinearSVC()`[7] function from the `sklearn.svm` class.

4.2.3 Gradient Tree Boosting (GTB)

Gradient Tree Boosting is an ensemble method that combines the output of several weak learners, i.e. weak models, to produce a larger, powerful model. The week models are decision trees of a fixed size because decision trees are particularly good at handling heterogenous data and modelling complex functions (scikit-learn: Machine learning in Python). The algorithm starts by fitting a first weak model $F_1(x) = y$ on

[7] `http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC` (accessed 5 April 2018).

the data. Then, the algorithm fits a next model on the residuals of the first model $h_1(x) = y - F_1(x)$. This new model $F_2(x) = F_1(x) + h_1(x)$ is combined with the new model in order to correct the errors, measured in the loss function L, of the old model (scikit-learn: Machine learning in Python). This step can be repeated, which makes Gradient Boosted Regression Trees build an additive model upon the initial function $F_1(x) = y$ with the task to find the model $h_1(x) = y - F_1(x)$ that reduces the loss function L at each step. This minimisation problem is solved numerically by searching for the steepest descent direction. The steepest descent direction is the negative gradient of the step size γ. Additionally, each learner is regularised by the shrinkage factor, also called learning rate, ν. As equation 4.4 shows, the learning rate strongly interacts with the number of weak learners. The smaller the learning rate, the more weak learners are required to achieve a constant loss function.

(4.4)

$$F_m(x) = F_{m-1}(x) + \nu\gamma_m h_m(x)$$

We ran GTB with the `GradientBoostingClassifier()` function from the `sklearn.ensemble`[8] module. We can choose between an exponential and a deviant loss function and set a number of other parameters. We decided to set the learning rate to 0.01, the number of weak learners to 150 and the maximum tree depth to 2 and remain with the default settings for the rest.

4.2.4 XGBoost

XGBoost is a scalable end-to-end tree boosting system developed by Chen and Guestrin (2016) that achieves state-of-the-art results in many machine learning tasks. While the underlying algorithm is similar to the Gradient Tree Boosting algorithm, as it is also an ensemble method that tries to minimise the loss function, it is modelled differently. They implemented a different regularisation term[9] (see shrinkage above) in order to control the complexity of the model and more successfully avoid overfitting (Chen and Guestrin, 2016, p. 786). However, the speed of the algorithm is probably the main reason, why many people use it. XGBoost makes use of parallelization, which means that it runs on all available cores. (Chen and Guestrin, 2016, p. 790).

[8] `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html` (Accessed 6 April 2018)

[9] For details, see Chen and Guestrin (2016).

Thus, we used the `XGBClassifier()` function from the `xgboost` python module. We set the learning rate to 0.09 and the number of weak learners to 300 and split CPU-usage to 8 different cores. The other parameters were set to default.

4.3 Feature Extraction and Selection

We extracted and tested a number of different features that have proven to be useful for the classification of word senses. As previously mentioned, lemma disambiguation is related to WSD, as it also tries to distinguish between different senses that are expressed in different lemmas. Hence, inspiration was found in Navigli (2009), who evaluated the performance of several traditional WSD systems based on a number of different machine learning algorithms and feature sets. The feature sets of successful systems contained context features, such as surrounding words and POS tags, syntactic features, such as syntactic cues and argument-head relations, topical features, which refer to the topic of the text and semantic features (Navigli, 2009, p. 10). We decided to test context words, POS tags and morphological analysis of the word in question as local features, argument-head relations as syntactic feature, word embeddings as semantic feature and translations as semantic-related feature. Additionally, we tested a global frequency feature. All the features were extracted as separate tabulator-separated stand-off annotations for each of the four languages, linked via the global token IDs of the ambiguous word forms. In the following, we will describe the features and if necessary additional preprocessing steps.

4.3.1 Local Features

4.3.1.1 Cooccurrence Features

The first local feature describes the context of the ambiguous words. We call it cooccurrence feature, as it describes the cooccurring words in a context window of $\{w_{-3}$ - $w_{+3}\}$ within a sentence. In order to reduce noise, we stripped punctuation and all stop words from the sentence before extracting the cooccurring tokens. For the stop word removal we used the Python `stop_words`[10] library, which features stop words lists in 22 languages, including the 4 languages with which we conduct our experiments. We also extracted the lemmas of the same context window, as we assume that a lemma-based approach could reduce sparse data, and thus improve performance for the 4 morphologically rich languages. The `.tsv`-files have all the same format, as presented in Figure 4.2. Column one contains the global Token

[10] `https://github.com/Alir3z4/python-stop-words` (Accessed 15 January 2018).

```
1446 Ehre Ehre NOUN NN Haus Präsidium wissen Freude mitwirken mein
1480 sie sie PRON PPER Politiker Ehefrau als beide viele unser
1510 Stärke Stärke NOUN NN Klugheit Rechtschaffenheit Beispiel
Gemessenheit abgeben dies
```

Figure 4.2: Sample of the extracted cooccurrence features for German.

ID, column two the word form, three the lemma and four and five the coarse- and fine-grained POS tag, while column six to ten contain the cooccurring word forms in ascending order. The first two columns render a scan of the entire corpus for each experiment unnecessary. The lemmatised cooccurrence feature has exactly the same format. In our experiments, we found that a context window of $\{w_{-2}$ - $w_{+2}\}$ performs better than the initially extracted window.

4.3.1.2 Morphological Analysis

The second feature is based on morphological information. Thus, we performed morphological tagging with 3 different systems on our four languages, depending on the availability of suitable models. We applied the models on previously extracted word forms in the required format. After the morphological tagging, we concatenated the word forms and the tagger output with the global token IDs, in order to create a morphological stand-off annotation for all four languages. We did not adapt the tag sets, as we only used this feature language-internally.

For German, we used the output of the *RFTagger* (Schmid and Laws, 2008), a Hidden-Markov-Model POS tagger, which is specifically built to deal with large tag sets with fine-grained tags, such as morphological tags. We tagged Italian and French with Morfette by Chrupala et al. (2008). The developers provide a model for French[11], and for Italian, we used the model trained by Baffelli (2016). For Finnish, we used *Finnpos* by Silfverberg et al. (2016), an open source morphological tagging a lemmatisation toolkit for Finnish[12]. *Finnpos* treats morphological tagging and lemmatisation as separate problems. Firstly, a CRF-classifier returns the tag sequences with the highest probability for a word sequence. In the following decoding stage, the classifier assigns the sequence with the highest probability and checks whether a morphological analysis is available in *OMorFi*[13] (Silfverberg et al., 2016, p. 870). *OMorFi* is finite-state transducer based morphological analyser by

[11] `https://github.com/gchrupala/morfette` (Accessed 22 February 2018).

[12] `https://github.com/mpsilfve/FinnPos` (Accessed 26 February 2018).

[13] `https://github.com/flammie/omorfi` (Accessed 26 February 2018).

Pirinen (2008), which delivers the lemmas for the sequences. If no lemma can be found in *OMorFi*, (Silfverberg et al., 2016, p. 870) employ the same method as the previously mentioned method by Chrupala et al. (2008). They treat lemmatisation as a classification task, with each class being a suffix edit script. We decided to test *FinnPos*, as the developers present significant improvements against other morphological taggers and it is open source with a pre-trained morphological tagging model for Finnish.

Eventually, we extracted all the token IDs with the corresponding morphological analyses from the stand-off annotations that refer to an ambiguous lemma or a lemma candidate.

4.3.1.3 POS tags

The third local feature encompasses the POS tags in a context window of $\{w_{-3}$ - $w_{+3}\}$. We extracted the fine-grained POS tags from the original CONLL-U-like corpus format, as we assume that the additional information in the fine-grained tag-sets could be relevant for the machine learning algorithm. As previously mentioned, these tag-sets differ considerably. However, it should not pose a problem, as we do not mix the disambiguation of the four languages. The four extraction files are identical with the ones for the cooccurrence words, except for that columns 6-10 contain the POS tags of the cooccurring words, instead of the word forms or lemmas. This results in four cooccurrence POS-tag files.

4.3.2 Syntactic Feature

We extracted argument-head relations of the target word and its head words from FEP6. Column 8 in the CONLL-U-like format contains the dependency relation of the target word and column 7 the ID of its head. Thus, we extracted the relation type of the target word, the relation type of its head and the relation type of the head of the head. The root of the dependency tree is marked as `root`. We stopped the extraction, when we reached the dependency root, i.e., the main verb. This results in small subtrees that may also contain information about words outside the direct context target word (Navigli, 2009, p. 12). We assume that the lemma candidates tend to maintain different syntactic functions in a sentence, which could be revealed by the small subtrees. The resulting `.tsv` files feature less columns as the dependency relation fit into columns 6-8. We can only extract dependency relations for German and Italian because the other two languages are not parsed.

```
70300 es es PRON _ _ _ _ _ _
70233 Fall Fall NOUN _ then__ADV _ affaire__NOUN _ tapaus__NOUN
70247 steht stehen VERB _ be__VERB evidente__ADJ apparaître__VERB
essere__VERB olla__VERB
```

Figure 4.3: Sample of the extracted translation features for German.

4.3.3 Cross-lingual and Semantic Features

This section discusses the semantic features that we tested in our experiments.

4.3.3.1 Translations

We extracted the translations of each target word as feature. Each lemma of a content word in the FEP6 is ideally aligned with a lemma of a content word in the other languages. However, sometimes no translation could be found, and not all of the words that we disambiguate are content words. In order to fill the gaps of the missing translations of content words, we chose the translations of four languages instead of just one. The more languages, the fewer gaps. We extracted the English and Spanish translation for all of the four languages that we want to disambiguate, as they both contain very few ambiguities. Additionally, we extracted German, French and Italian. Thus, we could extract the lemma[14], the language and the POS tag of the token for our experiments, as shown in Figure 4.3. Column 5 contains German, 6 English, 7 Spanish, 8 French, 9 Italian and 10 Finnish. The POS tag is directly attached to the lemma and can either be included or excluded in the feature.

We conducted several tests with the form in which we present the translation feature to the machine learning algorithm. We tested what happens if we kept the empty spaces in the source language. Sometimes, the machine learning algorithm tried to interpret the absence of the feature, which resulted in confusion and worse results. We also included Finnish as 5^th language, or reduced the amount of languages as features, which both decreased the performance.

4.3.3.2 Translation Feature Evaluation

We investigated whether there was one language in particular that has the biggest impact on the performance of the translation feature. This is interesting because

[14]We extracted the lemmas instead of the word forms in order to avoid sparse data through inflection and because the word-alignment was performed via lemma.

most language data is not readily available in a parallel multilingual environment[15]. In order to keep the expenses for future work small, we evaluated the most influential features of the multinomial Naive Bayes algorithm of the `scikit-learn` package by Pedregosa et al. (2011). The `MultinomialNB()` class has a built in `coef_` attribute, which "mirrors" the empirical logistic probability – see function 4.5 – of features cooccurring with a class[16].

(4.5)

$$\log_e P(x_i|y)$$

The `coef_` attribute of the `MultinomialNB()` contains as many feature rankings as class labels, unless it has only two classes. In this case, the rankings are flattened into one, with the features closer to zero describe a strong belonging to the first class, while the ones at the other end describe a strong dissociation, which can be translated into"belongs to the other class, namely class two". We then trained a model with the translation feature as sole feature. This feature contains 4 to 5 different languages, which are encoded in the feature names. Thus, we can map the features to the language and extract the top ten of the cooccurring features with each class label for each trained model and counted how often each language cooccurs as translation feature in the top ten for each language. The more a feature cooccurs with a certain class label, the more important it is for the Naive Bayes algorithm, considering that it is based on conditional probabilities. As raw counts are difficult to compare, we compare the proportion of top ten rankings for each language as translation feature in Table 4.2.

Apparently, German seems to be the best translation feature choice for French, Italian and Finnish, while English appears to be the best translation feature choice for German. The other options follow with quite a distance with Italian being second best translation feature choice for German, French and Finnish and French being second best choice for German. However, we have to keep in mind that these results also depend on word alignment, as well as lemmatisation quality, since word alignment was performed based on lemmas. German appears to have a high lemmatisation quality as previously revealed in the evaluation of the gold standard. Additionally, we do not know much about the word alignment quality in terms of

[15] We could generate translated data by deploying a machine translation system and translate the monolingual data and perform word alignment afterwards in order to extract the corresponding translations.

[16] `http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html` (Accessed 10 April 2018).

Data Language: \ Feature Language:	DE	FR	IT	EN	ES
DE	-	13.63 %	18.19 %	51.89 %	16.30 %
FR	48.53 %	-	19.67 %	15.44 %	16.36 %
IT	52.47 %	17.80 %	-	15.14 %	14.59 %
FI	41.60 %	13.29 %	16.26 %	15.11 %	13.74 %

Table 4.2: Proportion of top ten appearances per translation feature per language as feature performance measure. For absolute numbers, see Table A.1 in Appendix A.

correctness. Thus, the results could be skewed towards German.

4.3.3.3 Word Embeddings

This section discusses word embeddings as sole semantic feature that we tested in our experiments. Word embeddings are based on the distributional hypothesis (Harris, 1954), which claims that similar words occur in similar contexts. This idea is realised in word vectors, in which each target word $\vec{w_t}$ in a corpus V with a vocabulary size of $|V|$ is assigned a vector $\vec{v_w}$, which contains words in context c, represented as real numbers in a dense vector space. n is an arbitrary window size, which defines how many words to the right and left of the target word $|w_t|$shall be taken into account as context. As complex as this procedure may appear, an evaluation study of Iacobacci et al. (2016) shows that word embeddings lead to significant improvements of state-of-the art WSD with standard features. They combined standard features, such as cooccurring words and their POS tags with word embeddings of the context (Iacobacci et al., 2016, p. 904). They also tested word embeddings with different learning strategies and found *word2vec* to result in the best performing word embeddings (Iacobacci et al., 2016, p. 904). Hence, we will firstly introduce the toolkit, which we used for computing our embedding and then continue to describe their use as features.

We decided to use the *MultiVec* toolkit by Bérard et al. (2016) for calculating our word embeddings. *MultiVec* is a toolkit for computing word embeddings including Mikolov et al. (2013)'s *word2vec* features, Le and Mikolov (2014)'s paragraph features and Luong et al. (2015)'s model for bilingual word representations. The word representations are computed by a supervised machine learning approach with shallow neural networks. Shallow neural networks consist of an input layer, a hidden layer and an output layer. *MultiVec* has two different models at hand, which are based on Mikolov et al. (2013)'s *word2vec*. The continuous bag-of-words (CBOW)

model predicts the $\vec{v_w}$ based on the context. The skip-gram model on the other hand, does the exact opposite. It takes one vector, the vector of the target vector $\vec{v_w}$ as input layer and predicts the vectors of the context words in the output layer. The two models are represented in Figure 4.4.

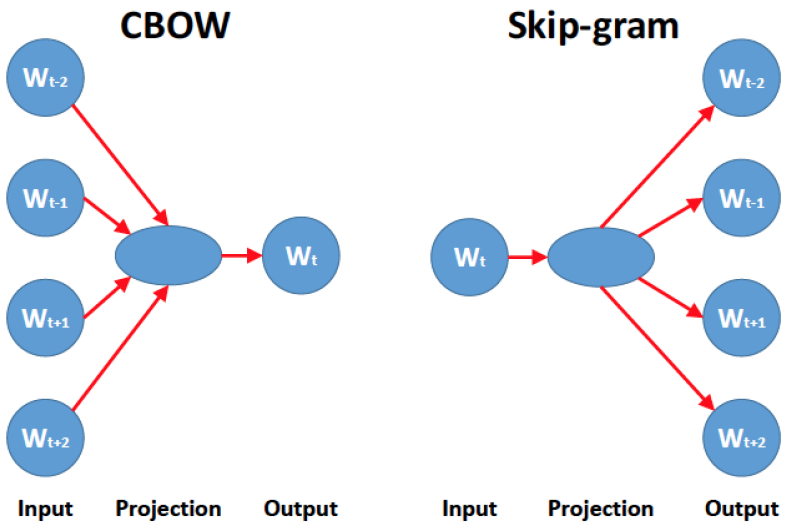

Figure 4.4: Representation of *word2vec*'s CBOW and skip-gram model as presented in (Bérard et al., 2016, p. 4189).

MultiVec does not only take into account data from one language, but from two languages at the same time. With the implementation of Luong et al. (2015)'s *bivec*, *MultiVec* can process parallel data and performs 4 updates at the same time, while *word2vec* performs one (Bérard et al., 2016, p. 4189). Firstly, it updates source language to source language, secondly source to target language, thirdly target to target language and lastly target to source. This results in bilingual word embeddings, which are projected into the same vector space, which makes direct translations appear very close to each other. *MultiVec* was tested against *word2vec* and *bivec* in different tasks and outperformed both models (Bérard et al., 2016, p. 4190–91).

We trained bilingual word embeddings with *MultiVec* in order to use them as features. We took each of the four languages for which we want to perform lemma disambiguation as source language and all the other languages as target language. We test the different combinations for best performance for each language. Additionally, *MultiVec* provides an evaluation script for English word embeddings, which we used for testing the semantic accuracy of the computed embeddings. On a first attempt, we computed embeddings with 256 dimensions, but soon realised that it was too many dimensions for our further processing with machine learning

algorithms. Thus, we chose 128 dimensions on a second attempt. We tested the skip gram versus CBOW model, context window sizes between 8 and 15 context words to the left and right and a negative sampling between 5 and 20. Additionally, we tested the amount of iterations necessary to deliver stable results. We found the skip-gram model to perform better than CBOW and a context window of 12 with a negative sampling of 15 with a minimum of 20 iterations to perform best. The resulting models are saved as binary files, which can be loaded directly from within any Python script in order to retrieve the vectors for any word in both languages. Thus, we generate the embeddings for each previously mentioned cooccurrence feature. The most straight forward way would be to concatenate the four word cooccurrence vectors. However, even if it is easy and straightforward, it does not necessarily yield good results. Taghipour and Ng (2015) found that word embeddings without dimension scaling do not perform well alongside other binary features. Thus, we implemented their proposed scaling function in 4.6 that scales each vector coordinate in a range of -1 to 1.

(4.6)

$$E_i \leftarrow \sigma \cdot E_i / stddev(E_i), i : 1, 2, ...d$$

E_i denotes the i^{th} dimension of the word embeddings matrix and σ is the target standard deviation. Several tests showed that 0.1 is a good value for the target standard deviation (Taghipour and Ng, 2015, p. 318). Thus, we scaled all word embeddings with the proposed scaling function with a target standard deviation of 0.1. Additionally, we averaged the context vectors instead of concatenating. The vector concatenation quadrouples the dimensions. This is unfavourable for machine learning, as high dimensional feature vectors require exponentially more memory. Thus, we decided to average the context vectors as proposed by (Iacobacci et al., 2016, p. 899), with the following function in 4.7.

(4.7)

$$e_i = \sum_{\substack{j=I-W \\ j \neq I}}^{I+W} \frac{w_{ij}}{2W}$$

The formula in 4.7 divides each dimension i of the vectors w_{ij} by $2\,W$, as the number of context words, W, is double the window size of the word embeddings, in order to compute the average of all embeddings. e_i then is a new vector with 128 dimensions with average values for all coordinates. As compared to concatenation

it uses significantly less memory and yields better results.

4.3.4 Feature Transformation

So far, apart from the word embeddings and the class weight feature, we have solely nominal features. This problem is traditionally solved via one-hot encoding because simply assigning different numbers to all the feature values will most likely cause any machine learning algorithm to infer a relational relationship between the different values (Raschka and Mirjalili, 2017, p. 116). This is not true for most nominal features, as for example the three values 'sing', 'choir' and 'church' for the feature cooccurrence do not have a highest or lowest value; they are simply different (Raschka and Mirjalili, 2017, p. 116). Therefore, it is advisable to apply one-hot encoding, a technique that creates a new feature for every feature value in the data set. The value 'sing' from the previously mentioned cooccurrence feature is then expressed as `[1, 0, 0]`, where the first entry is for 'sing', the second for 'choir' and the last for 'choir'. However, the drawbacks of one-hot encoding are that it contains relatively little information in much space and that it increases the length of feature vector by the amount *n-1* (n = amount of features. We transformed our nominal data into one-hot encoded feature vectors with the `sklearn.feature_extraction.DictVectorizer()` class[17].

4.4 Training

We train one classifier per ambiguous lemma and subsample the training set for frequent cases in order to save time and memory and to avoid learning problems whenever possible. Therefore, we filter the data accordingly. When loading the feature sets we filter the ambiguous lemmas with a look up in a dictionary that contains all ambiguous word forms and how often the corresponding lemma candidates occur independently in the corpus. This look up dictionary is created from files, which contain the word forms in the first columns, how often the word form occurs in the second column, the lemma candidates in the third column and the last columns[18] contain how often each lemma candidate occurs in the corpus, as shown in Figure 4.5. All the ambiguous word forms, with no evidence for one or both lemma can-

[17] `http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.DictVectorizer.html` (accessed 15 February 2018).

[18] Sometimes the tagger suggests just 2 lemma candidates for a certain word form, while it suggests up to five lemma candidates for other word forms. Thus the amount of columns, containing occurrence counts corresponds to the amount of lemma candidates and is individual for each word form.

```
abgeraten 6 abgeraten|abraten 0 43
abgewogen 89 abwiegen|abwägen 1 209
anführe 15 anfahren|anführen 11 1636
anführen 16 anfahren|anführen 11 1636
Angehört 1 angehören|anhören 1280 532
```

Figure 4.5: Excerpt from the German ambiguous lemma and unambiguously occurring lemma candidates counts.

didates are ignored, as they do not qualify for the distant semi-supervised machine learning approach. Word forms that feature the same lemma candidates such as *Armen* and *Arme* which can both be either 'arm' or 'pauper' are merged into the same set. The training set is sampled by extracting all the token IDs that belong to an independent lemma candidate. The token-ids can be mapped to the corresponding feature sets and the lemma candidate becomes the class label.

The subsampling is necessary because some lemma candidates occur several 10,000 times in the corpus and many classes are highly unbalanced as shown in Figure 4.5. Unbalanced classes pose a problem because many machine learning algorithms tend to become biased towards the majority class (Raschka and Mirjalili, 2017, p. 215). Additionally, more data does not always provide additional, necessary information. We have tested this by gradually reducing the maximum amount of training instances per lemma candidate. The results of the classifier's performance has stabilised with a maximum of 3,000 samples per lemma candidate. To obtain the maximally 3,000 instances, we have tested two subsampling methods. The first method subsampled the training set proportionally. This means that each class was reduced by the same factor in order to obtain less than 7,500 randomly drawn training instances in total. Whenever, one class would be reduced to zero instances, we kept one instance by default. The second method only draws random subsamples of classes that contain more than 3,000 instances and leaves classes with less than 3,000 instances untouched. While the first method more accurately reproduces the distribution of the original sample, the second method helps to reduce the majority bias. We had tested both methods, and decided for the method that only reduces classes that contain more than 3,000 instances, as firstly, the classifier performed better and secondly, we want our classifier to obtain useful information from our feature set.

4.5 Prediction

The distant semi-supervised version of our machine learning outputs predictions for all ambiguous word form/POS tag/lemma tuples, of which both lemma candidates occur in the corpus. All tuples with none or only one lemma part occurring in the corpus remain ambiguous. Thus the recall corresponds to the distribution of evidence for both classes, as presented in Table 4.1. The recall for German is 28 %, for French 66 %, for Italian 39 % and for Finnish 3 %. While some ambiguous lemmas, such as *überfahren|überführen* ('to overrun' and 'to convict' occur only once, others, such as *fallen|fällen* ('to fall' and 'to render' or 'to log') occur a thousand times and more. Nevertheless, we train a classifier for each instance, as we do not want to further reduce the coverage of this approach and the additional costs are low, once the pipeline has been established.

4.6 Testing

The predictions of the distant semi-supervised machine learning approach are tested against the entire larger part of the manually annotated gold standards. The setting does not require an n-fold cross validation, as we want the best machine learning algorithm to predict best on an entirely different data set. It is already in the nature of the set-up that training data does not coincide with test data. Thus, the algorithms are tested against the entire part A of the manually annotated gold standards, which were presented in Section 3.3. Nevertheless, including some data from the gold standard could improve the performance of the algorithm. The problems of this approach are that the test set becomes smaller and sparse ambiguous lemmas could entirely vanish from the test set. We test several feature combinations of the previously mentioned features with the machine learning algorithms introduced in section 4.2. The results of these tests will be presented in Chapter 6. The results and insights of the distant semi-supervised machine learning approach will also serve as foundation for the active learning approach, as we can hopefully determine insights on useful feature combinations and well-functioning machine learning algorithms for our problem.

5 The Active Learning Approach

Settles (2012, p. 53) defines active learning as the following: "[...] active learning, aims to improve upon supervised machine learning methods by making the most of the vast amounts of unlabelled data that may be available." This is done by "posing queries [to users] of the most informative instances", he continues. This chapter introduces the concept of active learning, the way we have integrated it into our lemma disambiguation pipeline, and how we group the ambiguous lemmas into different subgroups for informative queries.

5.1 Conceptualisation of the Active Learning Approach

We implemented an active learning pipeline in order to increase the coverage of our lemma disambiguation system, as a majority of ambiguous lemmas does not have evidence for all lemma candidates in the corpus. Active learning can close this information gap by asking the user to label unlabelled instances that the system decides to be useful (Pustejovsky and Stubbs, 2012). There are different methods on how the algorithm takes this decision and we decided to implement a density-weighted or pool-based method, i.e., an unsupervised learner that clusters our data into different groups, which we assume to be representative based on the features for our expected classes (Pustejovsky and Stubbs, 2012, p. 245). The idea behind this method, which is sometimes also called"optimal experimental design", is to gain as much information as possible with as less effort as possible (Settles, 2012, p. 5). However, the approach could be problematic because we only want to label a small amount of data, and it is yet uncertain whether this will suffice to train a new, precise classifier. On the positive side, we do know at least part of our pool, as we can extract the relevant samples by the ambiguous lemma. Thus, we just have to find instances that represent each of the different lemma candidates. Additionally, we can precompute the different samples, which saves memory and will make querying the user much faster.

Eventually, we developed a system that requests for user input, in cases where

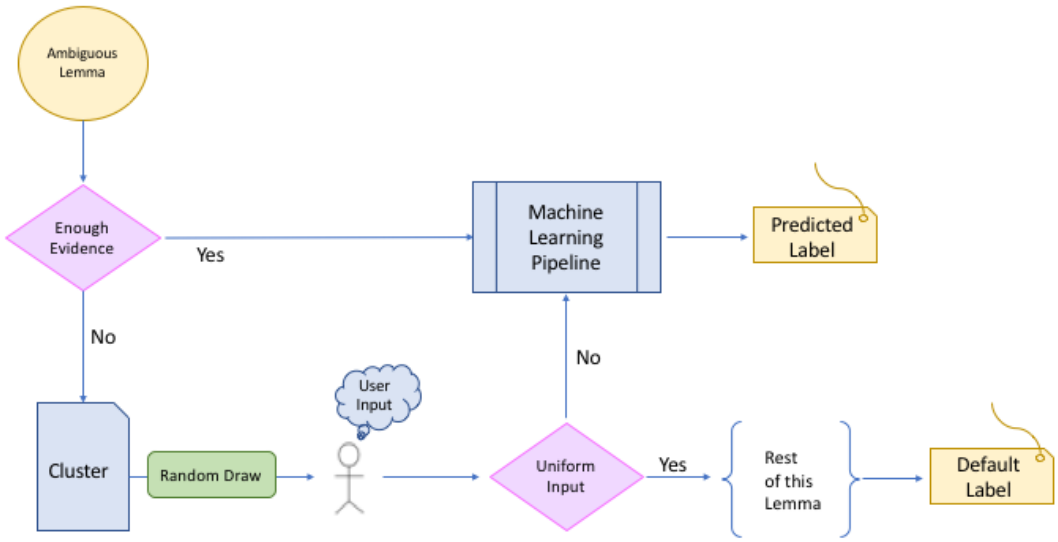

Figure 5.1: Graphic representation of our active learning pipeline. The "machine learning pipeline" refers to the pipeline represented in Figure 4.1 in Chapter 4.

we lack evidence for one or both lemma candidates as depicted in Figure 5.1. We load the precomputed clusters for the ambiguous lemma, group them into one additional cluster to the amount of suggested lemma candidates by the tagger and then draw a user pre-defined amount of instances from these clusters via token ID. With the token IDs, we query the sample sentence with the ambiguous word form from a pre-compiled SQLite database. The user then either decides for one of the lemma candidates suggested by the TreeTagger or enters another lemma. If all instances are assigned with the same lemma, we assign this lemma by default to all other ambiguous word forms with these lemma candidates. If the sample is assigned different lemma candidates a machine learning algorithm is trained with the user-labelled data as introduced in Chapter 4. We select the algorithm and feature combinations which performed best in the experiments in Chapter 4. While the clustering and sample draw is a mainly unsupervised procedure, the machine learning procedure itself is supervised, as the labels for the training data are obtained via human annotation. If we have enough evidence for both lemma candidates in the corpus, we proceed with the distant semi-supervised approach from Chapter 4. This pipeline enables a coverage of up to 100 % if the user is willing to also disambiguate lemmas that only occur once. Otherwise, the coverage depends on the user-set threshold for

the minimal occurrence for each ambiguous lemma.

5.2 Clustering

We precomputed clusters for all our ambiguous lemmas with insufficient evidence for our distant semi-supervised approach. The clustering was performed with the *fastcluster* package for hierarchical clustering by Müllner (2013)[1]. The main goal of clustering is to find subsets within all available observations that are interesting, homogenous and well separated from one another (Hansen and Jaumard, 1997, p. 191)[2]. The hierarchical clustering of *fastcluster* is based on dissimilarity matrices, which are computed from the original matrix, $X = N \times p$ for N being the number of observations and p being the number of features per observation. The dissimilarity matrix is computed as matrix $D = (d_{kl})$ of $N-1$ dissimilarities between the observations (Hansen and Jaumard, 1997, p. 193). The distance between two observations usually has one of the following properties: $d_{kl} \geq 0$, $d_{kk} = 0$ or $d_{kl} = d_{lk}$ (Hansen and Jaumard, 1997, p. 193). There are different ways, how these distances can be computed and the *fastcluster* package features the traditional Euclidean distance as well as other metrics[3]. Based on these initial distances, the cluster algorithm determines a pair of mutually closest points, i.e. a and b. These points are then merged into a new node n and the initial nodes a and b are deleted from the set. Afterwards, the algorithm needs to update the dissimilarity information of the new node n to all other nodes in the set. These steps are repeated until there is only one single node left (Müllner, 2013, p. 3). *fastcluster* provides seven different schemes – single, complete, average, weighted, ward, centroid and median – to update the dissimilarity of the clusters[4]. The output of the algorithm is a stepwise dendrogram, as depicted in Figure 5.2, encoded in a list of $N - 1$ triples, each containing information about which nodes are merged and their mutual distance. Other open source Python packages for hierarchical clustering are significantly slower and are suspected to also consume more memory (Müllner, 2013, 9).

[1] `http://danifold.net/fastcluster.html?section=0l` (Accessed 12 February 2018).

[2] In our specific case, we hope to find the observations belonging to our different lemma candidates in these clusters. It is, however, not guaranteed that the clustering reflects our criteria, as it is an unsupervised machine learning approach.

[3] For other options, as well as detailed explanations of the math behind, see Müllner (2017, p. 12).

[4] For a detailed description of these methods, see Müllner (2013, p.4).

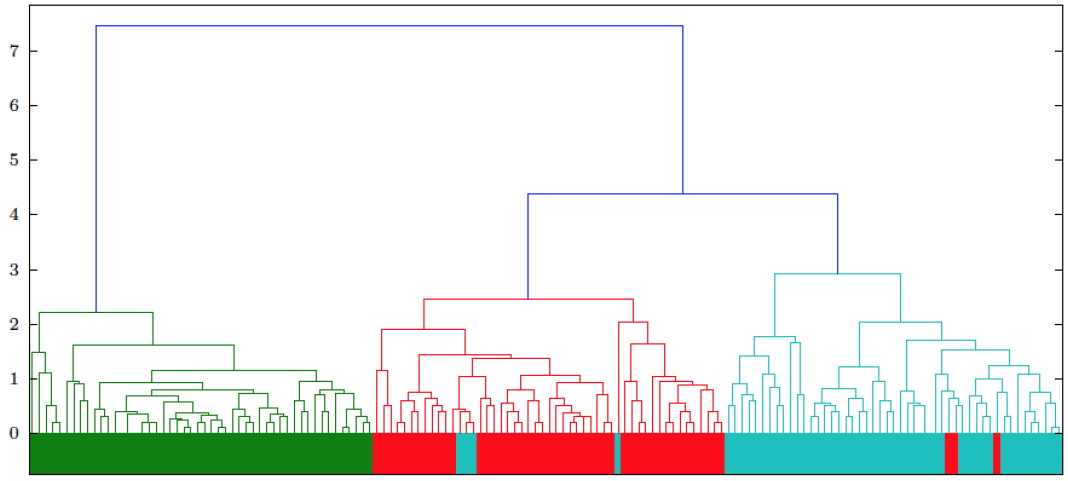

Figure 5.2: The output dendrogram of the hierarchical clustering as presented in Müllner (2013, p. 15).

5.2.1 Cluster Modification

We modified the output dendrogram of the hierarchical clustering to fit our purposes. Any unsupervised clustering algorithm returns an unpredictable amount of clusters; namely as many clusters as the algorithm computes to appropriately separate our observations. This is impractical for our purposes, as we want the number of clusters to represent the number of lemma candidates for each set of observations. We found an inexpensive cluster merging proposition by Piper (2012). He suggests to merge the clusters in the output dendrogram bottom-up, until one is left with the desired amount of clusters. Whenever we merge two clusters, we move one level up in the hierarchy; from the fine-grained clustering at the bottom, towards a more coarse-grained clustering towards the top (see Figure 5.2). We decided to merge the clusters until we are left with $n+1$ clusters, n being the amount of lemma candidates suggested by the TreeTagger. The additional cluster is for lemma candidates that could be present, but are unknown in the tagger lexicon. In that way, we take into account that some lemma candidates might be missing.

5.2.2 *fastcluster* Experiments

We tested four different distance update formulas with *fastcluster* and use the Cophenetic Correlation Coefficient (CCC) to measure and compare the quality of the output dendrograms. The distance update formula has a major influence on the

quality of the resulting clusters. Also important are the data structure and representation. *fastcluster* requires the input to be in a two-dimensional `numpy.array()`-format, containing feature vectors with floating point entries (Müllner, 2017, p. 8). The results from the experiments with the semi-supervised machine learning approach in Chapter 4 were chosen as starting points in order to determine the best features for the clustering algorithm. Fortunately, one of the best performing features were the numeric embeddings of the cooccurring token with the ambiguous word. In order to include the translations, we also calculated bilingual embeddings[5] with each of the translation languages. Thus, we could also represent the translations as numeric values. Aggarwal and Zhai (2012) wrote a survey on text[6] clustering in which they compared several clustering techniques, as well as different distance update formulae for hierarchical text clustering. They suggest the following formulae:

- **Single Linkage:** The Single Linkage distance update formula connects any pair of observations or clusters, so that their closest pair of observations have the highest similarity as compared to other observations or clusters (Aggarwal and Zhai, 2012, p. 91). This is an extremely fast and memory efficient method. Usually, it is just necessary to order the observations according to their computed similarity in descending order. However, this method can be problematic, as it can lead to "chaining". "Chaining" happens when the algorithm follows a chain of similar documents so that dissimilar observations end up in the same cluster (Aggarwal and Zhai, 2012, p. 91). X may be similar to Y and Y similar to Z, but this does not always mean that Z is also similar to X. We tested this method with the Python `fastcluster.linkage(x)` function.
- **Group-Average Linkage:** The Average Linkage formula considers the average dissimilarity of all the members of two clusters (Aggarwal and Zhai, 2012, p. 91). On the one hand, this method requires more memory and is considerably slower as the previous method, as it has to compute the distance between all members of two clusters. On the other hand, it yields better quality, as it is unaffected by the chaining issues. We tested this method with the `fastcluster.average(x)` function.
- **Complete Linkage:** As opposed to the Single Linkage method, the Complete Linkage method joins the clusters with the smallest maximum pairwise distance (Aggarwal and Zhai, 2012, p. 91). This method also avoids the chaining issue, as pairs of very dissimilar observations or clusters do not end up in the

[5]We also used the *multivec* package for this task.

[6]More precisely, they clustered documents. However, it was still a starting point.

same cluster (Aggarwal and Zhai, 2012, p. 91). We tested this method with the `fastcluster.complete(x)` function.

Additionally, we tested the Hamming method, as it is the only method that would allow us to work with one-hot encoded vectors. The Hamming distance equals the amount of differences at corresponding positions between two vectors of equal length (Müllner, 2017, p. 17). We tested this method with the general `fastcluster.linkage(x)` function by passing `hamming` as method argument. It is important to note that non-binary input would simply be one-hot encoded.

We tested these methods and features by calculating the average and median CCC for all clustered German lemmas with a lack of evidence in our corpus. The CCC was originally developed by Sokal and Rohlf (1962) for biostatistics and measures how adequately a computed cluster dendrogram preserves the initially computed pairwise distances. It has been applied to measuring clustering quality in various domains ever since. By calculating the CCC, we suppose that the original data $\{X_i\}$ has been been modified with a hierarchical cluster method to output a dendrogram $\{T_i\}$. We then define the distance measures between two observations i and j with the the original distance measures as $x(i,j) =\mid X_i - X_j \mid$ and the distance between the output model points of i and j as $t(i,j) =\mid T_i - T_j \mid$ (Toolbox, 2012). T always takes into account the output of the cluster model at the very bottom of the dendrogram, where the observations are first joined together. x is defined as the average of all $x(i,j)$ and $t(i,j)$, so that the CCC becomes c, as in 5.1 (Toolbox, 2012). The closer to 1 c is, the better the dendrogram preserves and represents the original distances.

(5.1)

$$c = \frac{\sum_{i<j}(x(i,j) - x)(t(i,j) - t)}{\sqrt{[\sum_{i<j}(x(i,j) - x)^2][\sum_{i<j}(t(i,j) - t)^2]}}$$

We used the `cluster.hierarchy.cophenet()`[7] function of the `SciPy` package for the implementation of the Cophenectic Correlation Coefficient. We computed the original Euclidian distances between the observations with the `scipy.spatial.distance.pdist()` function of the `SciPy` package. However, we could not calculate the Cophenectic Correlation Coefficient for all lemma pairs because the `pdist()` function consumes a lot of memory and we ran out of memory if we had too many observations[8] per

[7] `https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.cluster.hierarchy.cophenet.html` (Accessed 30 May 2018)

[8] We did not spend too much time on fixing the memory issue and just calculated the CCC for lemma pairs with $\leq 1,000$ observations.

Method	Score	Feature Type
Single Linkage	0.778	numeric
Group Average Linkage	0.865	numeric
Complete Linkage	0.753	numeric
Hamming	0.562	one-hot encoded

Table 5.1: Comparison of the average CCCs for German with the different distance update formulae.

lemma pair.

5.2.3 Cluster Quality

We chose the clustering method and feature combination that yields the highest average CCC for all lemma pairs. We started with the embeddings of the cooccurring words[9] with the exception of the Hamming method, for which we used the one-hot encoded vectors, and tested the four different distance update formulae mentioned above for German. We summarise the average CCCs in Table 5.1. The Group Average Linkage method yields the highest correlation. Thus, we added the embedding of the English translation as feature with the Group Average Linkage method, which increased the result to 0.988 for German. We tested whether the translation of an other translation language or all other languages would further increase the average CCC for all lemma pairs, but it was not the case. Adding, another language as translation feature decreased the average to 0.959 and adding all languages further decreased the value to 0.913, while also considerably increasing execution time and memory consumption. Thus, we just tested whether any other language as translation feature is better than English. The results vary within a range of 0.003, so we think that it is save to say that the embeddings of the cooccurring tokens together with the translation into one language yield the best results with respect to preserving the input distances of the observations, as well as in terms of speed and memory efficiency. This holds true for the other languages too, as shown in Table 5.2.

Besides the CCC for the quality estimation of the clusters, we compared the output for the user queries of 10 different ambiguous lemmas in German against a simple random draw, for which we assumed that both lemma candidates occur in our corpus. We wanted to verify that the feature and algorithm combination, which yields the highest CCC, really separates the data to our needs. We requested 15 user queries per lemma and and performed each request 4 times – twice with a draw from

[9]We used the same window, $w-2$ to $w+2$, from the distant semi-supervised classification.

Language \ Features	Embeddings	Embedding of 1 Translation
French	0.870	0.971
Italian	0.826	0.912
Finnish	0.826	0.914

Table 5.2: Average Cophenentic Correlation Coefficients for the other three languages with and without embedding for one Translation.

the precomputed clusters and twice with a random draw from all possible IDs. The results from this survey are summarised in Table 5.3, in which we list for each method how many examples we drew for the user prompts. The random draw method works well for cases such as *Messe|Messen* ('fair' and 'measuring'), *Beweis|Beweisen* ('proof' and 'proofing'), *Akt |Akte* ('act' and 'file'), *Pflanze|Pflanzen* ('plant' and 'planting') and *Pol|Pole* ('pole' and 'Pole'). Two of these pairs reveal to be resolved to the same lemma candidate all the time – *Pflanze* and *Beweis*. The other 3 cases might work well, as they probably have a more equal distribution of the two lemma candidates. For the other 5 cases the draw from the precomputed clusters works better, as this method is designed to capture potential minorities. The lemma pair *Spiel|Spielen** ('play', 'playing') is marked with an asterisk because we found a wrongly tagged instance of the verbal form *spielen* ('to play') that was capitalised at the beginning of a sentence. The precomputed clusters appear to help finding outliers and minority classes. The recognition of minority lemma candidates is crucial for exceeding a majority baseline. Even if the amount of outliers is so small that we manually disambiguate all of them, we profit from the fact that they are recognised and correctly resolved. Thus, we remain with the systematic random draw from clusters implementation.

5.3 Efficiency for User Queries

We created a SQLite database with example sentences for all instances of the ambiguous lemma pairs with a lack of evidence in the corpus in order to save memory and increase speed. Therefore, we decided to store the data in a SQLite database, which enables us to access just one chosen sentence at a time and discard it afterwards. We created the database by first extracting all the token IDs of the ambiguous word forms in the corpus, as well as word forms, of which one lemma candidate occurs less than 5 times. This enables an attempt to draw more observations of the minority lemma candidate, in order to improve the precision of

Lemma	1^{st} C	Freq.	Cl.	Freq.	Rd.	2^{nd} C	Freq.	Cl.	Freq.	Rd.
Bitte|Bitten	Bitte	8	10	11	14	Bitten	7	4	4	1
Messe|Messen	Messe	3	3	4	4	Messen	12	12	11	11
Besuch|Besuchen	Besuch	14	14	15	15	Besuchen	1	1	0	0
Beweis|Beweisen	Beweis	15	15	15	15	Beweisen	0	0	0	0
Pflanze|Pflanzen	Pflanze	15	15	15	15	Pflanzen	0	0	0	0
Akt|Akte	Akt	6	6	9	6	Akte	6	6	6	9
Herd|Herde	Herd	3	1	1	0	Herde	12	14	14	15
Spiel|Spielen*	Spiel	12	12	14	12	Spielen	3	2	1	3
Berg|Bergen	Berg	12	14	15	15	Bergen	3	1	15	15
Pol|Pole	Pol	4	3	9	6	Pole	11	12	6	9

Table 5.3: Comparison of training selection between draw from pre-computed clusters and the random draw method. C = Candidate, Freq. = Frequency, Cl. = Cluster, Rd. = Random.

the classifier. We extracted the word forms within their sentence context for those lemmas, which lack evidence for some lemma candidates, or are distributed very unevenly. The sentences were directly fed into the database with the word form in question highlighted with "***" for an easier and faster detection later in context by the user. This results in a simple and easily accessible database with one table with the token IDs as unique keys in the first column and the token context as field in the second column.

All SQLite procedures are executed with the Python built-in `sqlite3` module.

5.4 User Interaction and Input

The active learning pipeline lives from the interaction with the users and we accept user input at several points in the entire pipeline. Firstly, the users can decide on whether, they want to provide additional annotations for the system. If not, the system will just continue with the automatuc distant semi-supervised machine learning pipeline and disambiguate all the tokens that have evidence for both lemma candidates in the corpus. If the users choose to provide further input, they are prompted with two other options. So, they can decide on the minimum count, with which an ambiguous lemma occurs in the corpus, as well as the amount of word forms they are willing to disambiguate for each lemma pair. The minimal count for ambiguous lemma pairs has direct influence on the coverage of the system, as well as the time the users will spend on the disambiguation task. There is a considerable amount of lemma pairs (237 for German) that occur less than 10 times

in the corpus. However, we set 10 as the minimum of observation that the users have to disambiguate, as this is a really small amount of training data for a machine learning algorithm. This implies that the users would disambiguate all occurrences of the lemma pairs with less than 10 occurrences in the corpus manually. However, it would suffice to do this once, or maybe twice to reduce the error rate by validating the input, and rare lemma pairs are reliably resolved. Nevertheless, the users can choose to disambiguate either 10, 15 or 20 observations of the same lemma pair. This choice will not only influence the time they spend on the disambiguation task, but also the accuracy of the classifier output.

Secondly, the users will be prompted for the manual resolution of the ambiguous lemmas, which are randomly drawn from the saved clusters. When a cluster contains less than the number of lemma pairs to be disambiguated divided by the amount of lemma candidates plus one, we prompt for all instances in the cluster, as we assume them to be either rare cases or outliers. If the cluster contains more observations, we randomly draw the amount of lemma pairs to be disambiguated divided by the amount of lemma candidates plus one. Sometimes, this would result in less samples for disambiguation than the users chose to disambiguate. In these cases, we just randomly draw more items, as we assume to have found outliers already. These token IDs then serve as keys for queries in the previously mentioned SQLite database. We prompt the users with one sentence at a time, as depicted in Figure 5.3.

```
Choose between the two lemma candidates  Paar_1|Paaren_2  in the following example an
d enter the correct candidate:

Ich halte die Bestimmung , derzufolge nicht verheiratete Partner zugelassen werden kö
nnen , sofern in den Rechtsvorschriften des betreffenden Mitgliedstaats unverheiratet
e Paare mit verheirateten ***Paaren**** gleichgestellt werden , für etwas abwegig .
```

Figure 5.3: A sample user prompt for the correct lemma candidate for the word form *Paaren* ('pairing', 'couples').

The users are requested to choose between the proposed, numbered lemma candidates. As answer the system either accepts the the number for the correct candidate, or a string for cases, in which the correct candidate is not among the proposed lemma candidates. When the users input an empty string or a number outside of the range of lemma candidates, they will be prompted with the same request again, until they enter either a string or the number of a lemma candidate. This results in a list with token IDs and a list with resolved lemmas. If there are no word forms left for disambiguation, the token IDs and the resolved lemmas will directly be written to a file containing the results. Otherwise, they will be processed further.

So far, we have only created a simple terminal-based interface for our experiments. It is possible – and we recommend to do so – to create a more comfortable and user-friendly web-interface based on our pipeline.

5.4.1 Classification

We train a classifier for all lemmas, for which we have enough evidence for all lemma candidates in the corpus, of for which the users have found more than one candidate in the prompts. In cases, for which the users have selected the same lemma candidate for every prompt, we just assume that the majority hypothesis holds true and we assign the rest of all other observations with this lemma candidate and save the output in a file. This saves time and memory, and the draw from the clusters should ensure that any rare cases would have appeared in a cluster. A short analysis of the German, large gold standard set for the distant semi-supervised approach, shows that the distribution of a majority of 71 lemma pairs versus 19 lemma pairs is highly unbalanced, meaning more than 75 % of all lemma pairs are resolved to the same candidate. Hence, the loss by this default solution is expected to be small.

For the lemma pairs, for which the user selects various candidates, we train a classifier on the manual user-annotated data. Although, we have evidence for one or more lemma candidates for some lemma pairs, we decided not to include them into the training data. On the one hand, it would increase the majority class bias and on the other hand they contain false lemmas in cases, such as *Akt*|*Akte* ('act' and 'file'), for which every word form of *Akte*[10] can also be a word form of *Akt*, which is why we can only find evidence for the lemma *Akt* in the corpus. We rely on our results from the distant semi-supervised machine learning approach for the optimal choice of machine learning algorithm and the feature combinations. We also train a classifier for all cases for which we find evidence for both lemma candidates in the corpus in the same manner as in the distant semi-supervised machine learning approach. It thus turns out that the distant semi-supervised approach can be integrated in an active learning system, as outline in Figure 5.4.

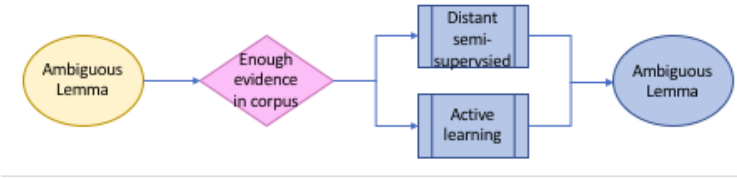

Figure 5.4: Graphic representation of the decision path within the full pipeline.

[10]To the best of our knowledge.

6 Results

In this chapter we present the results of the different experiments with our distant semi-supervised machine learning approach and our experiments with active learning.

6.1 Evaluation of the Distant Semi-supervised Machine Learning Approach

We have conducted several experiments with our distant semi-supervised machine learning approach. Firstly, we have performed an exhaustive search for the best feature and machine learning algorithm for German, before continuing our search more systematically. We also present our baseline, against which we measure our machine learning algorithms and the output from Graën (2018).

We chose precision as evaluation metric, as recall is stable for the distant semi-supervised machine learning approach because we can disambiguate all lemmas that have evidence for both lemma candidates in the corpus. Additionally, we want our system to deliver as precise results as possible. It is not only important that we can disambiguate as many lemmas as possible, but also that they are disambiguated correctly. There is an upper bound for all languages in part A of the gold standard because some lemma candidates are not included in the TT lexicon or assigned wrongly due to POS-tagging errors[1] It is impossible for the machine learning algorithm to learn them in the current set-up. Thus, we will present the results measured against the more benevolent upper bound, but declare the difference to the entire part A of the gold standard.

[1]The German word form *Bergen* can be either tagged as named entity and then it is unambiguous, or tagged as common noun and then it is ambiguous and can be either *Berg*|*Bergen* ('mountain' and 'salvaging'). In this case the name of the city and the nominalised verb form look alike, but the Finnish sentence initial *Jokin* tagged as pronoun is lemmatised to *some* and tagged as named entity to *Jok*|*Jokki* (Finnish personal names), which means that the tagger misses the correct lemma, although it is known.

Data set \ Language	German	French	Italian	Finnish
Upper Bound	64.3 %	65.5 %	64.5 %	70.1 %
Total	64.3 %	60.6 %	59.4 %	73.2 %

Table 6.1: Precision of Baseline for all four languages.

6.1.1 Baseline

We designed a baseline approach, with which we can compare performance of our distant semi-supervised machine learning approach. We decided for assigning the lemma candidate that occurs most frequently in the corpus to each ambiguous word form. Whenever all lemma candidates occur equally frequent, we make a random choice. This is an intuitive and fast method. The baselines for the four languages are presented in table 6.1. The precision of the baseline approach is well above 50 % for all languages. Finnish yields the highest results with a precision above 73.2 % against the upper bound of the gold standard.

6.1.2 Extensive Evaluation of German

We have performed an exhaustive search for the best feature combination and machine learning algorithm for German. We wanted to see whether there are any unexpected tendencies before performing a strategical search for all four languages. The results of the exhaustive search are displayed in Table B.1 in Appendix B. We tested 256 feature sets and machine learning algorithms in total. This large amount of tests lead to insights for our following procedures. We discovered that the XGBoost (Chen and Guestrin, 2016) is a misfit for our problem. XGBoost produced the worst results[2] in our experiments. Whenever we worked with one-hot encoded features[3], it started to produce results below our baseline as shown in Table 6.2. Additionally, XGBoost's hyper-parameters have to be tuned on the data set. We have a large number of data sets that we want to classify, therefore, tuning hyper-parameters for each data set is prohibitively expensive. Hence, we did not experiment with XGBoost on the other three languages. We also learned that combining a multitude of features does not necessarily yield better results. Thus, we

[2]Despite the outstanding performance of the XGBoost algorithm in many *Kaggle* machine learning competitions, including web text classification. For more information about *Kaggle*, see `https://www.kaggle.com/competitions`.

[3]This observation is confirmed by Brownlee (2016), who recommends XGBoost for "structured or tabular" data

Features	Embeddings	x								x	x	x		
	Majority Class		x											
	Cooccurrence T.			x							x		x	x
	Cooccurrence L.				x									
	Translations					x				x	x	x	x	x
	POS tags						x					x		x
	Morphology							x						
	Dependencies													
Algorithm	Baseline	65.1												
	Naive Bayes	+11.0	+8.9	+10.0	+10.0	+20.1	+10.3	+1.2	+9.4	+10.2	+17.8	+11.9	+18.8	+18.0
	SVM Linear	+12.7	+8.9	+9.4	+9.3	+20.1	+10.3	-1.5	+8.4	**+20.8**	+19.8	+18.8	+18.3	+18.8
	GTB	+9.6	+8.9	+8.5	+8.5	+17.0	+7.3	-1.7	+9.4	+16.7	+16.6	+15.8	+15.9	+14.8
	XGBoost	+11.4	+8.9	+2.5	+1.2	+5.7	+1.4	+1.5	+2.5	+15.7	+15.9	+15.8	-2.6	-8.5

Table 6.2: Improvements of precision of algorithms and feature combinations against the baseline in percentage points for German. 'T.' is for token and 'L.' for lemma.

decided to strategically combine those features that yield good results, when used alone and see how we can improve their performance.

Table 6.2 contains the single feature performance of the 4 tested machine learning algorithms, as well as the most promising feature combinations. The translation feature yields the best results with an increase in precision of 20.1 percentage points against the baseline with the linear SVM and the Naive Bayes algorithm among the single features, while the embeddings, majority class, dependencies and cooccurrence tokens and lemmas follow with an increase of roughly 12 percentage points with the linear SVM and Naive Bayes algorithm. We found the embeddings to yield the best results, when trained on German and French. As previously mentioned, XGBoost underperforms by far in most categories, apart from the word embedding feature, which is scalar. The linear SVM yields the best and most stable results in most cases, while the Naive Bayes yields very unstable results[4]. The Gradient Tree Boosting (GTB) yields average results for all features. We decided to present only the best performing feature combinations from our exhaustive search in Table 6.2. The linear SVM with the embedding and the translation feature yields the highest precision with 85.9 % against the upper bound and 84.8 % against the full part A of the gold standard.

6.1.3 Evaluation of French, Italian and Finnish

We systematically searched for the best feature and machine learning algorithm combination for the remaining three languages. From the exhaustive search for

[4]Before implementing the scikit-learn Bernoulli Naive Bayes algorithm, we had used the Multinomial Naive Bayes. The Multinomial Naive Bayes yielded better and more stable results, but is not compatible with scalar features that may contain negative values. Therefore, we had to switch to the Bernoulli Naive Bayes.

Features	Embeddings	x												x	x	x	x	x
	Majority Class		x								x	x	x			x		x
	Cooccurrence T.			x						x	x	x			x			
	Cooccurrence L.				x													
	Translations					x				x	x	x	x	x	x	x	x	x
	POS tags						x											
	Morphology							x				x	x				x	x
	Dependencies								x									
Algorithms	Baseline	65.5																
	Naive Bayes	+11.9	+15.5	+12.8	+12.8	+7.07	+10.7	+18.6	-	+17.2	+18.1	+16.8	+7.1	+8.4	+19.0	+7.96	+9.3	+8.5
	Linear SVM	+13.3	+17.7	+14.6	+13.3	+20.8	+12.8	+14.1	-	+18.6	+17.7	**+22.1**	+19.0	+19.0	+17.7	+21.2	+21.2	+21.7
	GTB	+16.4	+15.4	+19.9	+16.0	+16.8	+14.1	+21.2	-	+15.9	+17.2	+21.2	+21.7	+15.9	+15.5	+15.5	+15.9	+15.9

Table 6.3: Improvements of precision of algorithms and feature combinations against the baseline in percentage points for French. 'T.' is for Token and 'L.' for Lemma.

German we have learnt that it might work well to first run each feature individually and then combine the best performing features. We did not combine more than 4 features, as the evaluation of German had shown that more than four features still yield good results, but not better than having less features, while consuming considerably more memory.

The analysis of the single features in French reveals very fragmented results, which are summarised in Table 6.3. The morphology and cooccurrence token feature perform best by improving the precision by 19.9 and 21.2 percentage points respectively, but only with the GTB algorithm. The linear SVM yields an equally good result with the translation feature, by improving the baseline by 20.8 percentage points. The other features increase the baseline between 10-17 percentage points, with the embeddings, which perform best when trained on French and Italian, and the majority class being on the upper end. Therefore, we tested different combinations of embeddings, translations, cooccurrence tokens, majority class and morphology, while ignoring the differences of performance with respect to the choice of algorithm. The Naive Bayes still produces very unstable results when we combine the different features, while the linear SVM again produces stable and good results. The highest improvement is 22.1 percentage points against the baseline, yielded by the linear SVM with the majority class, the cooccurrence tokens, the translations and the morphology feature. The precision is 87.6 % against the upper bound and 82.7 % against the full part A of the gold standard.

Italian is the only language for which a single feature yields the highest overall improvement of the precision against the baseline, which comes at surprise. Table 6.4 shows that the linear SVM algorithm with the translation feature improves the baseline by 23.1 percentage points. Other well performing features are the cooccurrence tokens, the embeddings, the POS tags and the dependencies, which increase the precision by 14-17 percentage points. Further, we found the bilingual embed-

Features	Embeddings	x												x	x	x	x	x	x	x
	Majority Class		x																	
	Cooccurrence T.			x						x		x				x		x		
	Cooccurrence L.				x						x		x	x			x		x	
	Translations					x				x	x	x	x	x	x	x	x	x	x	x
	POS tags						x					x	x						x	
	Morphology							x												
	Dependencies								x					x				x	x	x
Algorithms	Baseline	63.5																		
	Naive Bayes	+16.9	+13.1	+15.7	+16.1	+10.8	+13.1	+7.6	+11.9	+17.9	+19.1	+20.6	+20.9	+20.8	+12.0	+18.3	+19.1	+19.6	+20.3	+16.6	+15.3
	Linear SVM	+17.5	+13.1	+15.6	+16.4	**+23.1**	+14.8	+2.9	+14.6	+22.7	**+23.0**	+21.0	+21.1	+22.3	+22.6	+18.2	+21.8	+22.2	+22.4	+20.6	+22.6
	GTB	+15.1	+13.1	+13.8	+15	+21.5	+13.6	+2.3	+16.2	+21.3	+21.3	+18.5	+18.6	+20.3	+20.5	+12.0	+20.2	+19.4	+19.3	+18.3	+19.6

Table 6.4: Improvements of precision of algorithms and feature combinations against the baseline in percentage points for Italian. 'T.' is for token and 'L.' is for lemma.

dings with German to perform best. None of the feature combinations with any algorithm performs as well as the translation feature on its own. However, we decided that it is not a good idea to rely on a single feature, despite the best results, as another data set might provide less information via translations. Therefore, we suggest to add the cooccurrence tokens, which results in an equally good performance, but still provides a back-off in cases, where the translation feature fails or is missing. Interestingly, combinations that consider the embeddings do not perform best. Their results are competitive, but do not exceed the performance of feature combinations without embeddings. The performance of the linear SVM algorithm is again the most stable, while the Naive Bayes again produces very unstable results (not shown in Table 6.4). The GTB algorithm produces good results, but not as good as the linear SVM. The linear SVM with the cooccurrence lemma and the translation feature performs best and yields a precision of 86.5 % against the upper bound and 82.5 % against the full part A of the gold standard.

Although the baseline performed best for Finnish, the machine learning results for Finnish do not exceed the results for the other languages. It just makes it harder to beat the baseline, as shown in Table 6.5. The best performing single feature is the embedding of the cooccurrence tokens with the linear SVM algorithm, which results in an increase in precision of 11.7 percentage points. We found the embeddings trained with Finnish and German to perform best. The cooccurrence tokens with the linear SVM algorithm perform equally good with an increase in precision of 11.5 percentage points. The Naive Bayes and the GTB do not perform as well. However, the performance of the GTB is nearly as good. We decided to combine the embeddings, the cooccurrence tokens, the cooccurrence lemmas and the majority class. Counterintuitively, the combinations containing the embeddings feature do not perform best. The best performance yields the cooccurrence tokens with the translation feature with or without the majority class feature by increasing the precision by 12.3 percentage points with the linear SVM algorithm. We prefer to

Features	Embeddings	x												x	x	x	x	x
	Majority Class		x									x	x	x				x	x
	Cooccurrence T.			x						x			x	x			x	x	
	Cooccurrence L.				x						x					x			x
	Translations					x				x	x	x		x	x	x	x	x	x
	POS tags						x												
	Morphology							x											
	Dependencies								x										
Algorithms	Baseline	73.2																	
	Naive Bayes	+8.9	+7.9	+7.9	+8.0	+9.1	+7.4	+7.4	-	+4.4	+9.0	-3.8	-8.7	+0.7	-2.1	+9.5	+3.3	+3.46	+8.8
	Linear SVM	+11.7	+9.8	+11.5	+8.8	+8.8	+7.1	+7.2	-	**+12.3**	+10.4	+8.3	+11.3	**+12.3**	+10.2	+11.3	+11.5	+11.2	+11.2
	GTB	+9.1	+7.9	+8.7	+7.2	+10.2	+7.4	+9.9	-	+10.1	+9.6	+10.1	+9.8	+10.1	+11.5	+10.7	+11.5	+10.9	+10.9

Table 6.5: Improvements of precision of algorithms and feature combinations against the baseline in percentage points for Finnish.'T' is for tokens and 'L' is for lemmas.

omit the majority class feature, as Finnish appears to have a strong majority class bias already[5], as we want the classifier to learn more than the most frequent class. This yields a final precision of 85.5 % within the upper bound and 84.4 % with the full part A of the gold standard.

6.1.4 Evaluation of Graën (2018)

We also evaluated the disambiguated data in FEP6 provided by Graën (2018), as we have been kindly provided with his data and he has evaluated a very small set of 53 instances in FEP9 that successfully yielded an unambiguous lemma[6]. We introduced his approach in Section 2.3 of Chapter 2. We chose precision, recall and the F_1-score as evaluation metrics and our part A of the gold standard for a comparison with our distant semi-supervised machine learning approach because both approaches require unambiguous evidence for all lemma candidates in the corpus. While our approach constantly reaches a recall of 100 %, the quantitative approach of Graën (2018) does not, because not all of the instances in the gold standard have matching word alignments. Additionally, we decided to compare the precision and recall within the upper bound, as both systems do not have the ability to exceed it.

Table 6.6 shows precision, recall and F_1-score for our distant semi-supervised approach compared to the approach of Graën (2018). Our approach outperforms the quantitative approach with the F_1-score being above 90 % for all four languages. The quantitative approach works best for German with an F_1-score of 87.6 %. Graën's (2018) quantitative approach performs slightly better in precision for German, French and Finnish, but not Italian. Finnish, yields the highest precision with 91.9 %, but also the lowest recall with 63.7 %. German and Italian yield the

[5]The baseline performance, which is based on the majority class, is above 70 % precision, which means that it is a good guess anyway.

[6]To our knowledge, the approach has not been evaluated for FEP6 so far.

Language	DE		FR		IT		FI	
Method / Metric	HE	GR	HE	GR	HE	GR	HE	GR
Precision	85.9 %	88.9 %	87.6 %	89.1 %	86.5 %	86.22 %	85.5 %	91.9 %
Recall	100 %	86.4 %	100 %	78.6 %	100 %	84.8 %	100 %	63.7 %
F_1-measure	**92.4 %**	**87.6 %**	**93.4 %**	**83.5 %**	**92.8 %**	**85.5 %**	**92.2 %**	**75.2 %**

Table 6.6: Performance comparison of the distant semi-supervised machine learning approach and the purely quantitative approach from Graën (2018). Column "HE" refers to our approach and column "GR" to Graën (2018).

highest recall for the quantitative approach with above 80 % . The major drawback of Graën's (2018) approach ist the low recall due to the lack of matching translations, which also leads to the lower F_1-scores. His approach would benefit from better word alignment quality.

6.2 Evaluation of the supervised active learning approach

We evaluated our supervised machine learning approach for German with the smaller part B of the gold standard, against a precomputed baseline. We also discovered a nice side effect of the user queries.

6.2.1 Baseline

Again, we assigned each occurrence of an ambiguous lemma with the majority class or a random choice of the lemma candidates if no majority class exists. For some lemma pairs, such as *Filter*|*Filtern* ('filter' and 'filtering') we find evidence for *Filter* in the corpus, so it is our majority class and we assign all cases if *Filter*|*Filtern* with *Filter*. This results in a precision of 55.1 % against the upper bound of the small gold standard and 54 % overall.

6.2.2 Evaluation of German

We evaluated the performance of our supervised machine algorithm with the smaller part B of the gold standard. The precision of our SVM model with the embedding and translation feature is 95.9 % against the upper bound of the gold standard and 93.9 % overall. However, the upper bound is relative in the context of active learning, as there is always a chance for finding at least one or all of the lemmas erroneously

assigned by the tagger during the user query phase. As we have answered the user queries ourselves, we also evaluated how many of the wrongly assigned candidates the algorithm should have known, as we, for example, manually disambiguated *Spiel|Spielen* ('play' and 'playing') to *Spiel* in 5 examples and *Spielen* in 10 examples, the classifier had a chance to learn the distinction between the two. However, there are also rare cases of the capitalised, sentence initial verb *spielen* ('to play'), which are tagged as noun and thus also assigned *Spiel|Spielen*. If such a case appears in the user queries, the classifier can learn to disambiguate it too, if it is not in the queries, it cannot. The classifier misclassified 6 word forms in total, of which 5 could have been resolved correctly, as we delivered the correct lemma candidate for another instance in the user queries. Indeed, the system asked for one of the candidates – *Bürgern* ('citizens'), which becomes *Bürge|Bürgen* ('bail' and 'bailing') due to a typographical error – that are not proposed by the tagger, which means that we had a chance to exceed the upper bound. Eventually, our system performs surprisingly good with respect to precision.

The recall of our model against the gold standard is 100 %. However, with respect to the entire corpus the coverage per ambiguous lemma types is 45.8 % because we miss 268 ambiguous due to the preset minimal lemma occurrence count of 15. Yet, this is a considerable increase as compared to the distant semi-supervised machine learning approach with a coverage of 28 %. Additionally, we could further increase the coverage anytime by re-running the process with a lower minimal count, which could be lowered as far as 1 occurrence which results in 100 % coverage[7].

We think that it is also important to evaluate the costs of the active learning process. By costs we mean the amount of time a user spends on disambiguating all the samples he is presented with. With 15 examples per lemma pair and a minimal lemma pair count of 16 it takes roughly 2.5 hours[8]. Lowering the minimal count to 1 would probably mean up to 5 hours or more for a user to work on the disambiguation task. While this costs are probably equally high or slightly lower for French and Italian, they are unproportionately higher for Finnish, which is the main reason why we would not necessarily recommend the active learning pipeline for Finnish.

6.2.3 Side Effects of Active Learning

Not only does the user input deliver correctly labelled data for the training of the machine learning algorithm, but it also delivers hints towards other errors in the

[7]This takes considerably more time.

[8]Note that we already had some experience, as we have tested this task multiple times for the detection of implementation errors and for debugging. Thus, it is very likely that it is more time consuming for unexperienced users.

data and corrects typos on the lemma level and provides more data for the gold standard. Firstly, the user prompts present the user a lot of context surrounding the ambiguous word form. This context is helpful to detect tagging errors, such as in example 6.1 of *Bergen*, a Norwegian city, which is wrongly tagged as common noun. This leads to the wrongly assigned lemma *Berg|Bergen* ('mountain' and 'salvaging'), as their common word forms coincides with name of the city *Bergen*. Such hints could be exploited in order to fix tagging errors and check other instances of *Bergen*.

(6.1) *Ein früherer Abgeordneter dieses Hauses, der von den*
a former member of this house who of the
Shetland-Inseln stammte, verwies gern darauf , dass von
Stetland Islands come.PAST point.PAST like out that from the
den Shetland-Inseln aus der nächst größere Bahnhof
Stetland Islands there the closest larger station the
der in Bergen ist.
in Bergen is
'A former member of this House, who came from the Shetland Islands, liked to point out that Bergen is the closest larger station from the Shetland Islands.'

Besides this, we also find capitalised verbs, such as *sagen* ('to say') and *gestalten* ('to shape') in sentence initial position, which are wrongly POS-tagged as nouns. Thus, they are assigned the lemma of the nouns, which are by chance ambiguous. Lastly, the user prompts also help in detecting typographical errors. The word form *Bürgern* ('citizens'), for example, is misspelled to *Bürgen* ('bailing') almost systematically. Out of the 15 instances of *Bürgen*, it should be *Bürgern* 7 times. Hence, we do not only provide input for the supervised machine learning algorithm, but also detect erroneous data in the corpus.

Lastly, all the manually annotated data can be added to enlarge the gold standard. We could even compare the input of several users and only accept the input as golden that has been given at least twice.

7 Conclusion

This thesis is an endeavour devoted to the investigation and amelioration of standard lemmatisation practices with a focus on ambiguity resolution. So far, relatively little has been done in this field. This can probably be attributed to the fact that English has a tiny amount of ambiguous word forms. The lemmatisation tools, such as LEMMING, Morfette and Lematus presented in the related works may well outperform the TreeTagger (TT) in terms of accuracy. However, pre-trained models are often unavailable in non-mainstream languages, due to the lack of adequately annotated language resources. Additionally, they may even be trained on TT output, such as Lematus and thus simply reproduce the ambiguities of the TT. Thus, our approach of trying to ameliorate the TT output in multiple languages seems to be legitimate, if not necessary. We searched for inspiration in the neighbouring field of the more fine-grained task of word sense disambiguation, and found several approaches that work with multilingual data.

We chose German, French, Italian and Finnish as languages of interest, as we found them to have a large number of ambiguities in the *Full European Parliament Corpus v.6*, a large multilingual parallel corpus. We have worked with a pre-release version in a CONLL-U-like format with linguistic annotation, including TT POS-tags, lemmas and dependency relations. We found the TT output for Finnish to have the largest amount of ambiguous unique types of word form/POS tag/lemma tuples. German shows more than 10-times less ambiguous unique word form/POS tag/lemma tuple types, followed by French and Italian. In French and Spanish, the phenomenon is almost absent.

Further, we developed a gold standard for German, French, Italian and Finnish. We manually annotated German and French and engaged other linguistic experts for manually annotating Italian and Finnish. The gold standards served for the evaluation of our models, and are available online[1] as resources for others who pursue research in the field of lemma disambiguation. The manual evaluation of the gold standard data also helped us to estimate the precision and recall of the tagger for word forms, for which the TT suggests two or more lemma candidates. Additionally,

[1] `https://gitlab.cl.uzh.ch/jasminheierli/gold-standards-lemmadesambiguierung` (Accessed 1 June 2018).

we tried to examine the coverage of the TT by applying GerTwol on our German corpus, respecting the TT POS-tags. While the TT appears to achieve almost 100 % coverage for adjectives, roughly a third of the examined nouns and verbs received an additional lemma candidate.

When it comes to the automatic disambiguation of the lemmas, we developed two different models. Firstly, we developed a distant semi-supervised machine learning algorithm that makes use of independently occurring lemma candidates in the corpus. Thus, we do not need any manually annotated data for training, as we find the training data directly in the corpus. We assumed that the translations of these lemmas would be very informative for the algorithm. This assumption was confirmed in all four languages. Additionally, we tested traditional contextual features, auch as cooccurring tokens and POS tags, as well as more modern features, such as word embedding with different machine learning algorithms. We found the linear SVM models to perform best for all languages and yield best precisions above 85 %, which also beat our baseline in all four languages. The results of our models are competitive with the quantitative multilingual approach of Graën (2018) in terms of precision and oupterfom it in terms of recall and F_1-score. However, the example of Finnish clearly illustrates the limits of our first approach. The coverage is only 3 %, as a vast majority of lemma pairs lacks evidence, as unambiguous lemmatisations, for one or more lemma candidates in the corpus. On the other hand, the coverage for the other three languages is at least 10 times higher.

Secondly, we developed an active learning pipeline that helps to overcome the knowledge gap by prompting the user with carefully selected ambiguous items lacking evidence for at least one lemma candidate in the corpus. Then, the user either disambiguates the lemma by specifying the number of the correct candidate, or entering another suggestion, if the correct lemma is not among the propositions from the TT. The samples for the user are drawn from precomputed and modified clusters, in order to ensure that minority classes and outliers are found and included in the training data. We tested the pipeline for German and achieved a precision of 95.9 % against the upper bound[2] and an overall precision of 93.9 %. Despite the promising results, we would like to point out that selecting the correct lemma is time consuming. Thus, we implemented a threshold for the minimal occurrence of ambiguous lemmas in the corpus. Our presented results are calculated with a threshold of 16 occurrences. However, the threshold does not only save time, but also lowers the coverage. For German, we achieved a coverage of 45.8 % of all word form/POS tag/lemma tuple types within the corpus with the threshold. Lastly, the

[2]The upper bound does not consider ambiguous lemmas assigned with a lemma that is not suggested by the tagger – earlier reffered to as false negative – because they are unobtainable for machine learning algorithms.

manual annotation task leads to additional insights into our data. We are likely to discover POS-tagging errors and typographical errors that can at least be corrected on the lemma level.

Eventually, the two approaches can be combined[3] and lead to a quite reliable resolution of roughly 50 % for languages with tantamount of evidence for individual lemma candidates in the corpus.

7.1 Future Work

The example of Finnish demonstrates that for some languages the approach needs to be further adjusted. We discovered that some lemma pairs, such as *laajentuminen#neuvottelu|laajentua#neuvottelu* ('enlargement negotiations' and 'enlarge negotiations') of the compound word form *laajentumisneuvotteluissa*, only the first part of the compound appears to be ambiguous and we also found lemma pairs that only differ in the second part. Therefore, one could try to find evidence for the subparts of the lemma candidates and if available exploit these for the classifier training.

Additionally, we could experiments with different algorithms and features. So far, we have neglected the field of Neural Networks, which are currently exploited for many NLP tasks, including lemmatisation, as presented by Bergmanis and Goldwater (2018). We also believe that we can achieve more with the word embeddings. We have only tested one approach in order to scale and adjust them for a machine learning algorithm, and we are aware that it probably was not the best with respect to expected performance. As we have the resources to work with word embeddings for the translations of the lemmas in question, we could also experiment only with word embedding features.

Due to time restrictions, we could only properly evaluate the active learning pipeline for German. Nevertheless, it would be interesting to see how far we would come with the same approach for at least French and Italian. Additionally, we could expand the active learning queues for highly imbalanced classes, in order to search more training samples for the minority class. We assume that this would further improve the performance of the machine learning approach. And yet, the results we were able to obtain are promising and lead to the assumption that the performance of the models could be further improved by taking such measures.

[3]We actually did this in the implementation of the active learning pipeline as it did not cause additional efforts. We already had the implementation of the distant semi-supervised approach.

References

C. C. Aggarwal and C. Zhai. *A survey of text clustering algorithms*, chapter 4, pages 77–128. Springer, 2012.

C. Baffelli. An annotation pipeline for Italian based on dependency parsing. ma-thesis, University of Zurich, June 2016.

T. Bergmanis and S. Goldwater. Context sensitive neural lemmatization with Lematus. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1391–1400, 2018. URL `http://homepages.inf.ed.ac.uk/s1044253/papers/Context_Sensitive_Neural_Lemmatization_with_Lematus.pdf`.

J. Brownlee. A gentle introduction to XGBoost for applied machine learning, Aug. 2016. URL `https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/`.

A. Bérard, C. Servan, O. Pietquin, and L. Besacier. Multivec: a multilingual and multilevel representation learning toolkit for NLP. In *The 10th edition of the Language Resources and Evaluation Conference (LREC 2016)*, 2016.

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

G. Chrupala, G. Dinu, and J. van Genabith. Learning morphology with Morfette. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, pages 2362–2367. European Language Resources Association (ELRA), May 2008.

J. Cohen. A coefficient agreement for nominal scales. *Educational and Psychological Measurement*, (20):37–46, 1960.

D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, Apr. 1992.

M. Diab and P. Resnik. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 255–262. Association for Computational Linguistics, 2002.

A. R. S. Dimitar Kazakov. Using parallel corpora for word sense disambiguation. In *RANLP*, pages 336–341, 2013.

J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 363–370, 2005.

W. A. Gale, K. W. Church, and D. Yarowski. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26(5):415–439, 1992.

H. Glück and M. Rödel. *Metzler Lexikon Sprache*. J.B. Metzler, 5. edition edition, 2016.

J. Graën. Identifying phrasemes via interlingual association measures - a data-driven approach on dependency-parsed and word-aligned parallel corpora. In C. Konecny, E. Autelli, A. Abel, and L. Zanasi, editors, *Lexemkombinationen und typisierte Rede im mehrsprachigen Kontext*. Stauffenburg Verlag, 2017.

J. Graën. *Exploiting Alignment in Multiparallel Corpora for Applications in Linguistics and Language Learning*. PhD thesis, Universität Zürich, 2018.

J. Graën, D. Batinic, and M. Volk. Cleaning the Europarl corpus for linguistic applications. In *Konvens 2014*, 2014.

M. Haapalainen and A. Majorin. Gertwol: ein system zur automatischen Wortformerkennung deutscher Wörter. Technical report, Lingsoft, Inc., Sept. 1994.

P. Hansen and B. Jaumard. Cluster analysis and mathematical programming. *Mathematical Programming*, 79(1-3):191–215, 1997.

Z. S. Harris. Distributional structure. *Words*, 10(2-3):146–162, 1954.

I. Iacobacci, M. T. Pilehavar, and R. Navigli. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 897–907, 2016.

D. Jurafsky and J. H. Martin. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. Third Edition draft, Aug. 2017. URL `https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf`.

P. Koehn. Europarl: A parallel corpus for statistical machine translation. *Machine Translation Summit*, 5, 2005.

Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1188–1196, 2014.

E. Lefever and V. Hoste. Semeval-2010 task 3: Cross-lingual word sense disambiguation. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, DEW '09, pages 82–87. Association for Computational Linguistics, 2009.

E. Lefever and V. Hoste. Parallel corpora make sense: Bypassing the knowledge acquisition bottleneck for word sense disambiguation. *International Journal of Corpus Linguistics*, 19(3):333–367, 2014.

P. Liang, B. Taska, and D. Klein. Alignment by agreement. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 104–111, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

E. Loper and S. Bird. Nltk: The natural language toolkit. In *Proceedings of the 2nd ACL Workshop on Elective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, volume 1, pages 63–70. ACL, 2002.

T. Luong, H. Pham, and C. D. Manning. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159, 2015.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of English: The Penn Treebank. *Compuational Linguistics*, 19(2): 313–330, June 1993.

T. McEnery, A. Wilson, and F. S.-L. Amalio. Multilingual resources for European languages: contributions of the crater project. *Literary and Linguistic Computing*, 12(4):219–226, 1997.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

T. Müller, R. Cotterell, A. Fraser, and H. Schütze. Joint lemmatization and morphological tagging with Lemming. In *EMNLP*, pages 2268–2274, 2015.

D. Müllner. fastcluster: Fast hierarchical, agglomerative clustering. *Journal of Statistical Software*, 53(9):1–18, 2013.

D. Müllner. *The fastcluster package: User's manual*. The Comprehensive R Archive Network, `https://cran.r-project.org/web/packages/fastcluster/vignettes/fastcluster.pdf`, Aug. 2017.

R. Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41 (2):1–69, Feb. 2009.

H. T. Ng, B. Wang, and Y. S. Chan. Exploiting parallel texts for word sense disambiguation: An empirical study. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 455–462. Association for Computational Linguistics, 2003.

J. Nivre and M. Ballesteros. Maltoptimizer: An optimization tool for maltparser. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 58–62. Association for Computational Linguistics, 2012.

J. Nivre, J. Hall, and J. Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, pages 2216–2219, 2006.

F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

N. Ott. Evaluation of the bananasplit compound splitter. techreport, Seminar fur Sprachwissenschaft, Eberhard-Karls-Universitat Tubingen, Mar. 2006.

E. Parliament. European Parliament - never lost in translation. web, Oct. 2008. URL `http://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//TEXT+IM-PRESS+20071017FCS11816+0+DOC+XML+V0//EN#title1`.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay. Scikit-learn:

Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

S. Petrov, D. Das, and R. McDonald. A universal part-of-speech tagset. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*, pages 2089–2096, 2012.

J. Piper. Simple hierarchical clustering in python 2.7 using scipy. `https://warwick.ac.uk/fac/sci/sbdtc/people/students/2010/jason_piper/code/`, Feb. 2012. Accessed 20 March 2018.

T. A. Pirinen. Automatic finite state morphological analysis of Finnish language using open source resources (in Finnish). Master's thesis, University of Helsinki, 2008.

A. Przepiórkowski, R. L. Górski, B. Lewandowska-Tomaszczyk, and M. Łaziński. Towards the national corpus of Polish. In *The Proceedings of LREC 2008*, pages 827–830, 2008.

J. Pustejovsky and A. Stubbs. *Natural Language Annotation for Machine Learning: A Guide to Corpus-building for Applications*. O'Reilly Media, Inc., 2012.

S. Raschka. *Python Machine Learning*. Packt Publishing, 2015.

S. Raschka and V. Mirjalili. *Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-learn, and TensorFlow*. Packt Publishing, 2nd edition, 2017.

B. Santorini. Part-of-speech tagging guidelines for the penn treebank project (3rd revision). *Technical Reports (CIS)*, 1990.

A. Schiller, S. Teufel, and C. Stöckert. Vorläufige Guidelines für das Tagging deutscher Textcorpora mit STTS (Draft). Technical report, Universität Stuttgart, Institut für maschinelle Sprachverarbeitung, 1995. URL `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/stts_guide.pdf`.

H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Natural Language Processing (NeMLaP)*, 1994.

H. Schmid and F. Laws. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *Proceedings of the 22nd*

International Conference on Computational Linguistics (COLING 2008), pages 777–784, Manchester, Great Britain, Aug. 2008.

scikit-learn: Machine learning in Python. 1.11.4. Gradient Tree Boosting. http://scikit-learn.org/stable/modules/ensemble.htmlgradient-tree-boosting.

R. Sennrich, O. Firat, K. Cho, AlexandraBirch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. Läubli, A. V. M. Barone, J. Mokry, and M. Nadejde. Nematus: a toolkit for neural machine translation., Mar. 2017. URL `https://arxiv.org/pdf/1703.04357.pdf`.

B. Settles. *Active Learning*, volume 6:1 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers, 2012.

M. Silfverberg, T. Ruoklainen, K. Lindén, and M. Kurimo. Finnpos: an open-source morphological tagging and lemmatization toolkit for Finnish. *Language Resources and Evaluation*, 50(4):863–878, Dec. 2016.

R. R. Sokal and F. J. Rohlf. The comparison of dendrograms by objective methods. *Taxon*, 11(2):33–40, 1962.

A. Stein. French TreeTagger part-of-speech tags. online, Apr. 2003. URL `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/french-tagset.html`.

K. Taghipour and H. T. Ng. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 314–323, 2015.

M. S. Toolbox. Cophenetic correlation coefficient. `https://link.springer.com/content/pdf/10.1186%2F1029-242X-2013-203.pdf`, 2012.

D. Varga, P. Hal'acsy, A. Kornai, V. Nagy, L. Németh, and V. Tr'on. Parallel corpora for medium density languages. In *Proceedings of the Recent Advances in Natural Language Processing (RANLP)*, 2005.

M. Volk. Choosing the right lemma when analysing German nouns. In *Multilinguale Corpora: Codierung, Strukturierung, Analyse. 11. Jahrestagung der GLDV*, Frankfurt, 1999.

M. Volk, C. Amrhein, N. Aepli, M. Müller, and P. Ströbel. Building a parallel corpus on the world's oldest banking magazine. In *Proceedings of the 13th*

Conference on Natural Language Processing (KONVENS 2016), pages 288–296, 2016.

A. Voutilainen, K. Muhonen, T. K. Purtonen, and K. Lindén. Specifying treebanks, outsourcing parsebanks: Finntreebank 3. In *Proceedings of LREC 2012 8th ELRA Conference on Language Resources and Evaluation*, 2012.

I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier Inc., 3rd edition, 2011.

D. Yarowski. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics, 1995.

A Tables

Feature Language: / Data Language:	DE	FR	IT	EN	ES
DE	-	260	347	990	311
FR	264	-	107	84	89
IT	1,421	482	-	410	395
FI	2,883	921	1,127	1,047	952

Table A.1: Absolute numbers of top ten appearances per translation feature language per language as performance measure.

B Figures

Algorithm / Feautres	naive Bayes	SVM Linear	Boosted Trees	XGBoost
ectr	74.53	83.18	80.47	79.73
ecoltr	83.06	84.18	79.48	79.97
ecotr	82.08	84.05	80.84	80.22
etr	74.53	**84.92**	81.01	79.97
eco	74.23	75.28	75.03	74.66
e	75.28	77	73.92	75.65
c	73.18	73.18	73.18	73.18
co	74.29	73.67	72.8	66.75
col	74.29	73.55	72.8	65.51
tr	84.42	84.42	81.33	69.96
pos	74.54	74.54	71.57	65.64
m	65.51	62.8	62.55	65.76
d	73.67	72.68	73.67	66.75
ccotrm / all	83.55	78.12	69.96	62.17
ccotr	83.44	82.57	80.22	61.68
ccom	74.28	66.13	64.18	56.86
cotrm	84.42	78.37	69.96	60.94
ctrm	82.94	74.04	69.96	61.68
ctr	84.79	83.56	81.45	61.68
cco	74.41	73.61	72.8	64.52
cm	65.51	61.8	62.55	65.76
cotr	**83.06**	82.57	80.22	61.68
com	72.06	66.38	65.39	58.84
trm	82.45	74.29	70.21	60.44
ccol	74.41	73.79	72.8	64.52
coltr	85.04	82.57	80.47	61.68
colm	72.06	66.38	64.15	58.84
coltrm	84.42	78.37	69.96	60.94
ccolmtr	83.56	78.12	69.09	60.94
ccolm	74.28	66.01	64.16	56.86
ccoltr	85.04	82.57	80.35	61.68
cotrpos	82.32	83.06	79.11	55.75
coltrpos	82.13	82.89	79.32	56.25
cotrd	83.31	82.94	**80.96**	57.23
ccotrmposd	82.08	78.37	71.07	51.3
coltrd	83.19	**83.07**	**80.84**	57.23
ccoltrmposd	82.2	78.37	70.83	51.3
ccoltrm	83.55	78.12	69.96	62.17
ccolposd	74.54	67.99	65.39	48.7
coltrmpos	81.83	80.52	71.24	51.34
coltrmposd	82.2	78.62	70.95	49.57
ccoltrmd	82.07	77.26	69.47	56.98
ccoltrmpos	81.82	78	70.7	54.02
ccoltrposd	81.58	83.06	79.72	56.48
ccoposd	75.77	76.02	74.41	53.4
cotrmposd	82.2	78.61	71.57	49.57
ccotrmd	82.07	77.26	70.33	56.98
ccotrmpos	81.82	78	70.7	54.02
ccotrposd	81.83	73.67	79.73	52.16
posd	74.29	75.4	73.55	60.69
mposd	68.36	66.01	62.91	53.52
cposd	74.53	75.15	73.67	58.59
trposd	84.18	83.68	79.73	53.77
coposd	76.27	75.77	74.41	55.99
colposd	76.01	75.9	74.66	55.99
copos	76.27	75.52	72.68	60.69
colpos	76.14	75.4	73.05	60.69
cod	75.15	73.68	74.9	59.58
cold	75.28	73.61	75.03	58.34
mpos	66.74	63.91	63.16	57.48
md	66.99	62.79	61.68	58.22
trmd	82.7	74.91	69.47	55.5
cmd	68.11	63.16	62.29	58.22
cmpos	67.37	63.91	63.16	57.72

Figure B.1: The results of the exhaustive best feature and machine learning algorithm search for German.

C Lemmatisation Guidelines

Lemmatisierungsrichtlinien

Danke, dass du dir die Zeit nimmst, mich bei meiner Masterarbeit zu unterstützen!

Du hilfst mir dabei einen Goldstandard, um die Genauigkeit meines Machine Learning Algorithmus an manuell annotierten Daten zu messen. Meine Classifier lernen, aufgrund von grossen Mengen vorverarbeiteter Sprachdaten mehrdeutige Wortformen einem Lemma eindeutig zuzuordnen. Dies passiert über diverse mathematische Berechnungen. Um herauszufinden, wie gut sie mit Rechnen sind, benötige ich von Hand korrigierte Daten; diese stellst du mir nun her. Meine maschinell annotierten Daten werden danach in einem weiteren Schritt mit deinen manuell annotierten Daten verglichen und ich bekomme so eine Idee, wie gut oder schlecht meine Classifier arbeiten.

Die Daten sind in einer Excel-Datei aufbereitet. Du arbeitest Zeile um Zeile nach unten. Die erste Spalte beinhaltet eine ID, die für dich nicht weiter relevant ist. In der zweiten Spalte siehst du die Wortform, wie sie im Originaltext vorkommt. In der dritten Spalte siehst du das mehrdeutige Lemma. Ein Lemma ist die Grundform eines Wortes; so wie sie im Wörterbuch stehen würde. Wir Menschen können aufgrund des Kontextes meist sofort bestimmen, welches der beiden Lemmata korrekt ist. Die Maschine kann das aber nicht ohne weiteres. In der vierten Spalte ist der Vorschlag eines Classifiers. In der fünften Spalte kommt nun dein Einsatz: wenn das vorgeschlagene Lemma korrekt ist, trägtst du ‚1' ein, wenn es falsch, aber in Spalte 3 enthalten ist, trägst du 0 ein, und wenn das richtige Lemma weder in Spalte 3 noch in Spalte 4 vorkommst, trägst du deinen eigenen Vorschlag ein. Die Entscheidung kannst du aufgrund des Kontextes der in Spalte sechs bis acht gegeben ist treffen.

roire|croître	croître	✗	0	La convergence économique n' aurait jamais été	crue	possible encore , dans cette ampleur , il y a un an .
roire|croître	croître	cru		Cette catégorie couvrirait les fruits et les légumes	crus	, certains produits laitiers tels que le yaourt , et quelques bières .
roire|croître	croître	cru		Pour notre part , nous avons toujours défendu la logique de la vérité , aussi	crue	soit -elle .
roire|croître	croire	✓	1	Si nos parents et nos grands-parents avaient dit « ne vous inquiétez pas , vos petits-enfants siégeront ensemble au sein d' un parlement d' une Europe unie » , personne ne les aurait	crus	.

Im obigen leider etwas verpixelten Bildausschnitt siehst du einen Ausschnitt aus meinem Dokument für Französisch.
In der obersten Zeile hat der Classifier ‚croître' vorgeschlagen. Das ist falsch, denn im Kontext rechts ist klar ‚croire' richtig und somit habe ich den Classifier mit 0 bewertet.
Beim zweiten und dritten Beispiel hat der Classifier wiederum ‚croître' vorgeschlagen. Hier stimmt aber weder ‚croître', noch ‚croire' aus Spalte 3. Hier ist ‚cru' vom Adjektiv ‚cru' abstammend, also habe ich das so eingetragen.

Im Finnischen hat es manchmal ‚#' in den einzelnen Lemmata. Leider, weiss ich nicht, was sie zu bedeuten haben. Vielleicht sagen sie, aber dir etwas. Wenn du deren Bedeutung kennst, oder sie erahnen kannst, bin ich froh darum, wenn du deine Gedanken mit mir teilst.