A practitioner's view: a survey and comparison of lemmatization and morphological tagging in German and Latin

Rüdiger Gleim[1], *Steffen Eger*[2], *Alexander Mehler*[1], *Tolga Uslu*[1], *Wahed Hemati*[1], *Andy Lücking*[1], *Alexander Henlein*[1], *Sven Kahlsdorf*[1], *and Armin Hoenen*[1]
[1] Text Technology Lab, Goethe University Frankfurt, Germany
[2] Ubiquitous Knowledge Processing Lab, Technische Universität Darmstadt, Germany

ABSTRACT

The challenge of POS tagging and lemmatization in morphologically rich languages is examined by comparing German and Latin. We start by defining an NLP evaluation roadmap to model the combination of tools and resources guiding our experiments. We focus on what a practitioner can expect when using state-of-the-art solutions. These solutions are then compared with old(er) methods and implementations for coarse-grained POS tagging, as well as fine-grained (morphological) POS tagging (e.g. `case`, `number`, `mood`). We examine to what degree recent advances in tagger development have improved accuracy – and at what cost, in terms of training and processing time. We also conduct in-domain vs. out-of-domain evaluation. Out-of-domain evaluation is particularly pertinent because the distribution of data to be tagged will typically differ from the distribution of data used to train the tagger. Pipeline tagging is then compared with a tagging approach that acknowledges dependencies between inflectional categories. Finally, we evaluate three lemmatization techniques.

Keywords: morphological tagging, lemmatization, morphologically rich languages, NLP evaluation modeling

1 INTRODUCTION

Lemmatization and part-of-speech (POS) tagging are critical preprocessing steps for many natural language processing (NLP) tasks, such as information retrieval, knowledge extraction, and semantic analysis. In morphologically rich languages such as German and Latin, both processes are non-trivial due to the variability of lexical forms. This results in large tagsets for both coarse-grained and fine-grained (morphological) POS tagging – including inflectional categories such as `case`, `gender`, and `degree` in addition to coarse-grained POS labels – and a large number of (potentially unseen) forms associated with each lemma. In this work, we survey tagging and lemmatization techniques for German and Latin, using corpora that allow us to analyse the effects of NLP between genres (Tiger vs. TGermaCorp), and also between periods (Capitularies vs. Proiel). Our survey includes both older tools, such as the TreeTagger (Schmid 1994) and TnT (Brants 2000), and more modern approaches to tagging and lemmatization. Even though we expect technology to improve steadily over time, it is not always easy to quantify the gap between older and more modern approaches, or to rank the most recent generation of systems in order of efficiency. We test our systems under the following conditions and requirements:

- We train lemmatization and tagging independently because the methods studied are designed for separate use.[1] We then focus on the impact of varying parameters, supplementary resources, and a combination of tools.
- Ideally, we want a learned system to perform well on the data on which it has been trained (*in-domain* (ID): a specific text genre, historical language variant, etc.) but also to perform adequately on similar corpora (*out-of-domain* (OD): with similar but different genres, registers, language varieties, etc.).
- Since coarse-grained POS tagging alone may be insufficient for linguistic applications and unsatisfactory for practitioners, we expect a system to perform reasonably well on fine-grained POS tagging.

[1] LemmaTag (Kondratyuk *et al.* 2018) is an exception that natively supports joint lemmatization and tagging.

- As run-times of systems may be of considerable interest for practitioners, we include both training and testing time estimates for each technique.

Section 2 provides a systematization grid for NLP applications and their evaluation. This grid is mapped on to our evaluation objectives, thereby offering both a general evaluation model and a roadmap for the subsequent experimental sections. In Section 3, we describe three approaches to lemmatization, followed in Section 4 by ten tools for tagging, which form the basis of our experiments. Section 5 introduces all resources used in the experiments, primarily the corpora used for training and evaluation, with a discussion of other lexical resources and methods of computing word embeddings. Section 6 describes the experiments used to test lemmatization and POS– as well as fine-grained POS tagging. In Section 7, we discuss our findings, while Section 8 provides a summary of this study and prospects for future work.

2 NLP EVALUATION ROADMAP

Taggers cannot be compared or even applied *in vacuo*; the minimum requirements for NLP taggers are a tagset and a target text of some natural language. Similar dependencies apply to virtually all NLP applications. In order to systematize such relationships and make them transparent for readers of NLP-related work, we provide an interrelationship model in Figure 1. The tree structure on the left-hand side of the model presents NLP tasks and resources, followed by instantiating parameters, which are then mapped on to evaluation objectives. The dashed lines in Figure 1 indicate partial use of a parameter set with respect to an objective. Thus, the systematization grid explicates the requirements to be met in order to perform NLP experimentations. Accuracy values for such experiments assess how adequately relevant objectives have been attained. These objective-to-accuracy mappings constitute the right-hand side of Figure 1 (column "A", values given in %). We chose the maximum accuracy achieved as target value for the scale (which usually lies in the range between 96% and 98%). We also indicate which section presents the relevant evaluation studies. The systematization grid thus presents a general dependency model of NLP tasks and resources, as a roadmap for the current paper.

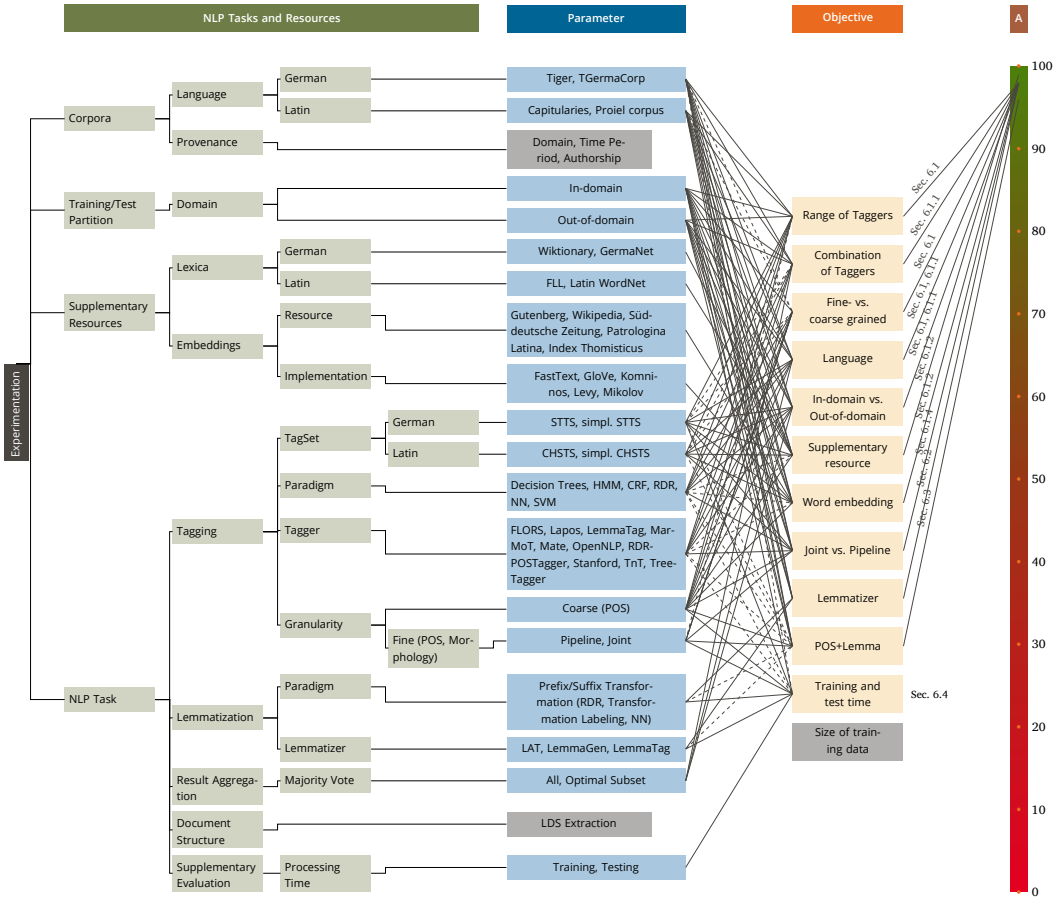

Figure 1: Roadmap for NLP evaluation. See main text for details. The tree on the left-hand side provides a systematization of tasks and resources for NLP. The leaf nodes ("Parameter" column) collect the parameter settings we employ to instantiate the relevant cells of the systematization grid. Parameters that are not part of the current paper are shaded in gray. The parameters are mapped in different ways on to the evaluation objectives, as indicated by connecting lines. Dashed lines indicate that the parameter instantiation in question is not fully exhausted by the target evaluation objective. The scale on the right-hand side depicts the maximum accuracy of the evaluation results for each objective, and indicates the section where the relevant evaluation study is described

3 LEMMATIZATION

We view *lemmatization* as the problem of transforming a word form into its canonical form, or lemma. In a machine-learning context, lemmatization has sometimes been considered as a character-level string transduction process (Dreyer *et al.* 2008; Nicolai *et al.* 2015; Eger 2015; Schnober *et al.* 2016), a prefix and suffix transformation problem (Juršič *et al.* 2010; Gesmundo and Samardzic 2012), or as a pattern-matching task (Durrett and DeNero 2013; Hulden *et al.* 2014).[2] While character-level string transducers may yield excellent results (Nicolai *et al.* 2015), particularly when trained and tested on lists of words randomly extracted from a lexicon (Eger 2015), they tend to be learned slower, and typically do not lemmatize in context, but consider the lemmatization problem in isolation, ignoring contextual word-form cues. In addition, we found in preliminary experiments that, for real-world lemmatization, where distribution is marked by many irregular forms, simpler prefix and suffix transformation systems may be competitive with more sophisticated string transducers.

In this work, we experiment with three approaches to lemmatization, two of which are based on prefix and suffix transformations, and one on neural networks. These experiments are presented in Section 6.2. **LemmaGen** (Juršič *et al.* 2010) learns 'ripple down rules' (Compton and Jansen 1988), that is, tree-like decision structures, from pairs of strings. Rule conditions are suffixes of word forms, and rule consequents are transformations that replace the suffix in question by a new suffix. The second approach we experiment with is the casting of lemmatization as a classification task (Gesmundo and Samardzic 2012), which we call **LAT**: lemmatization is viewed here as a 4-tuple indicating the prefix and suffix transformations involved in the lemmatization process. For example (see Figure 2), the transformation of the German verb form *gespielt* into its lemma *spielen* is encoded by the

[2] Lemmatization can also be implemented with the help of a lexicon. However, lexicons are hard to acquire, and their performance is comparatively low in cases where they do not sufficiently discriminate the distributions of polysemous lemmas in large corpora of real texts. Nevertheless, a lexicon could typically 'assist' a learned system, e.g., via features that trigger if a form occurs in the lexicon (e.g., in a similar manner to that outlined here).

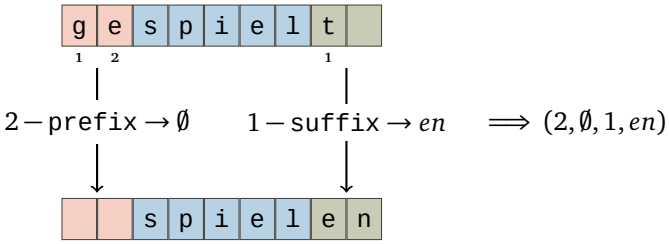

Figure 2: Example of how to represent lemmatization for *gespielt* to *spielen* as a tuple

tuple $(2, \emptyset, 1, en)$, indicating that to derive *spielen*, the first two characters of *gespielt* are replaced by the empty string, and the last character is replaced by *en*. This compact encoding considers lemmatization as a classification problem where the size of the output space is relatively small (some hundreds or thousands of labels, at most). Moreover, lemmatization can then also be treated as a sequence labeling problem, where dependency between subsequent labels may be taken into account. One may argue that inflections in German are rich and that modifications may also include central characters, thus replacing the entire word in extreme cases. Nonetheless, this approach can be applied as long as the tagger can handle the output space.

Finally we include LemmaTag (Kondratyuk *et al.* 2018), which is based on neural networks. It has primarily been included in experiments for POS tagging (see Section 4), but it is also interesting to work with as it supports joint lemmatization and tagging.

4 POS TAGGING

Among the milestones in POS tagging (or sequence labelling) are: including *dependencies* between output labels (as in Markov models such as HMMs or CRFs); the broad use of lexical *features* (Ratnaparkhi 1996; Toutanova *et al.* 2003); and the concept of the *margin* introduced in SVMs. The most recent class of taggers is characterized by several possibilities: that of including *word representations* learned from *unlabeled* data, that of applying feature-rich models to problems with large output spaces, and that of making use of *deep* (rather than *shallow*) models such as neural networks that can in addition function without handcrafted features.

In this work, we consider the following POS tagging systems, which are listed in order of their year of publication. The **TreeTag-**

ger (Schmid 1994), a popular tagging system until recently, is based on decision trees. As such, it cannot account for dependencies between output (tag) labels. **TnT** (Brants 2000) implements a trigram Hidden Markov tagger with a module for handling unknown words. It has been shown to perform as well as maximum entropy models. The **Stanford tagger** (Toutanova *et al.* 2003) implements a bidirectional log-linear model that makes broad use of lexical features. The implementation lets the user specifically activate and deactivate desired features. **Lapos** (Tsuruoka *et al.* 2011) is a 'history-based' tagging model (this model class subsumes maximum entropy Markov models) incorporating a lookahead mechanism into its decision-making process. It has been reported to be competitive with globally optimized models such as CRFs and structured perceptrons. **Mate** (Bohnet and Nivre 2012) has been introduced as a transition-based system for joint POS tagging and dependency parsing. We also include the **OpenNLP** tagger, an official Apache project.[3] For these systems, we refer to the original works for more in-depth descriptions.

Among the most recent generation of taggers, we consider **MarMoT** (Müller *et al.* 2013), which implements a higher order CRF with approximations such that it can deal with large output spaces. In addition, MarMoT can be trained to fire on the predictions of lexical resources as well as on *word embeddings*, real vector-valued representations of words.[4]

The **RDRPOSTagger** (Nguyen *et al.* 2014) implements an error-driven approach to POS tagging by constructing a tree of single classification ripple down rules (SCRDR). It takes a gold standard *and* an automatically tagged version thereof as input in order to generate rules to reflect any differences. By default, RDRPOSTagger uses a built-in lexicon-based tagger, which by itself is not very accurate, but learning exception rules from the initial tagging gives promising results. Note that, since the approach is based on the concept of correcting the output of some initial tagging, the initial tagger is required for both training and testing/tagging. Nguyen *et al.* (2016) show that using an external tagger (e.g. TnT) over the built-in (lexicon-based)

[3] See `https://opennlp.apache.org/`.

[4] Another morphological tagger which is based on conditional random fields is TLT-CRF (vor der Brück and Mehler 2016) which we could not include due to time and space constraints.

tagger yields better results. In this work, we replicate their experiment by optionally using TnT, and extend it by using MarMoT as the initial tagger. Note that, in order to achieve a tagged version of the training set for our experiments, we use 10-fold jack-knifing: each fold is tagged by TnT or MarMoT based on the remaining folds. Taking a practitioner's view, we use the built-in tagger for most of our experiments. However, we also examine the benefits of using an external tagger for POS.

FLORS (Schnabel and Schütze 2014) tags a given word by constructing a feature vector representation of its local context and then classifying this vector by an SVM. The feature vector representation of each word in a context includes distributional, shape, and suffix information, and the feature vector for the entire context is the concatenation of the word vector representations. Note that the implementation of FLORS includes language-specific features (for English). This is expected to decrease its performance on the Latin and German datasets we consider. In principle, the vector representations of words are the same for known and unknown words, thus FLORS is potentially very well-suited for OD tasks. In our work, we use online FLORS (Yin *et al.* 2015), which incrementally updates word representations for each new test sentence encountered.

We also wanted to include **NonLexNN** (Labeau *et al.* 2015), a non-lexicalized neural network architecture for POS tagging. By operating on the subword/character-level, it promises to yield higher performance on OD tasks, similarly to the FLORS tagger. However, we could not make this tagger perform on-par with the other taggers surveyed. One reason for this was its very lengthy runtime – several days for a single training fold – so that we could not sufficiently experiment with its parameters.

A neuronal network approach based on bidirectional long short-term memory recurrent neural networks (Wang *et al.* 2015) was implemented using Deeplearning4j. It consists of one Graves Bidirectional Long-Short Term Memory (BLSTM) Layer, and one RNNOutputLayer with a Softmax activation function and MultiClass Cross Entropy (MCXENT) Loss function. The BLSTM Layer uses a TANH activation function, and we changed the updater to ADAGRAD, to improve learning for rare POS tags. Following Wang *et al.* (2015), we represent each word with its word embedding, and add a three-dimensional binary

	User-def. features	Large output spaces	External resources	Label dependencies
FLORS		✓	✓	
Lapos				✓
LemmaTag		✓		✓
MarMoT	✓	✓	✓	✓
Mate		✓		✓
OpenNLP		✓		✓
RDRPOSTagger		✓		✓
Stanford	✓			✓
TnT		✓		✓
TreeTagger		✓		

Table 1: Systems and selected properties

vector to retain information about capitalization: initial upper case, all upper case, or all lower case. We add another vector of 3 dimensions to identify the word ending. Note that this approach could not be fully exploited during our experiments, because of limited hardware resources and the amount of time needed to train the networks.

Finally, we include **LemmaTag** (Kondratyuk *et al.* 2018), a featureless neural network approach to joint lemma and POS tagging, based on bidirectional memory recurrent neural networks. It uses character- and word-level embeddings. LemmaTag is based on Tensorflow, a library for dataflow programming widely used in machine learning applications. Since it benefits from GPUs, we run LemmaTag on a GPU workstation so as to run all experiments in reasonable time.

In Table 1, we list some key properties of the taggers in the survey. While most models make use of features (except for HMMs as TnT is based on, for which the inclusion of arbitrary features is non-trivial), not all of them allow users to specify user-defined features. Therefore, we had to exclude Lapos and Stanford from some experiments (e.g. joint-tagging) as they do not scale well on large output spaces.

In addition to the list of taggers, we include a **majority vote** POS tagger. By examining the results of at least three taggers, we identify the POS tag for a given token with maximum tagger agreement. In order to solve ties, taggers are ranked by date of publication. In practice, tagger accuracy should be estimated on a held-out set, extracted from the training data, or on a (small) hand-annotated data set. The majority vote tagger is used to assess whether tagging system errors

correlate. The assumption is that majority voting does not work when all systems commit the same types of errors.

5 DATASETS

In this work, we examine the performance of NLP tools on Latin and German texts. We distinguish between in-domain (ID) and out-of-domain (OD) experiments. From a machine learning perspective, ID experiments are more well-defined, because they are generally performed on a corpus of texts from the same era and genre. More importantly, the gold standard of such corpora has usually been created by a closed group of trained annotators who agreed upon a specific annotation manual. Thus, we can expect a high degree of coherency in terms of the primary content as well as the annotation.

For ID experiments the data must be partitioned into distinct training and test sets. In general, we perform a 3-fold random subset validation, with a 90%/10% split on each corpus for each language. In contrast, OD experiments involve two corpora, with one corpus used entirely for training, and the other one for testing. For the LemmaTag tagging and lemmatization tool, we require an additional development set. For these ID experiments, we use a 3-fold 80%/10%/10% split. For the OD experiments, we use 90% of the entire source corpus as the training set and the remaining 10% as the development set. As before, the entire target corpus is used for testing and the 90%/10%-split is performed three times. Since OD experiments bring together corpora that may vary in terms of the era in which they were written, in the genres they cover, and in the standards by which they were annotated, we expect a significant drop in accuracy when we evaluate NLP tools, as has been documented by much previous research (Müller *et al.* 2015; McGillivray *et al.* 2009). Nonetheless, OD scenarios provide a much more realistic perspective on the performance of lemmatization and POS tagging, since a practitioner usually has to rely on pre-trained models. This is not primarily because of the technical skills required to train a model, but rather because of the huge effort required to construct a training set large enough to accurately cover the desired genre.

In the following subsections, we describe the corpora as well as supplementary resources used in this work for German and Latin. Of

Corpus	Language	Sentences	Tokens
Tiger	German	50,472	888,238
TGermaCorp	German	8,941	157,210
Capitularies	Latin	15,572	481,578
Proiel	Latin	1,147	22,280

Table 2:
Statistics of corpora used in the experiments

the taggers we consider, MarMoT can benefit from additional lexical resources as well as word embeddings. In order to examine the impact of supplementary data, we evaluate various lexical resources and different corpora, as well as algorithms to compute word embeddings.

5.1 *Corpora*

For German, we train and test on the Tiger corpus (Brants *et al.* 2004), and on TGermaCorp (Lücking *et al.* 2016). The Tiger corpus consists of newspaper articles from the German "Frankfurter Rundschau" which were semi-automatically lemmatized and tagged for POS and morphology. For Latin, we use the Capitularies corpus (Mehler *et al.* 2015; Eger *et al.* 2015) and the Proiel corpus (Haug and Jøhndal 2008). The Capitularies corpus is based on the "Capitvlaria regvm Francorvm, ed. Alfredus Boretius (Hannover 1897)". The two Latin corpora stem from different genres *and* different epochs, making them interesting candidates for OD tagging experiments. The Capitularies consist of instructions and directives from the Merovingian and Carolingian periods (600–900 AD), whereas Proiel consists of classical and Christian texts (100 BC–500 AD). We use a random subset of Proiel, for which tag labels have been manually synchronized with those of the Capitularies. As the Proiel corpus does not contain punctuation, we expect low accuracy when it participates in OD scenarios. This problem should not occur with the Capitularies corpus, which contains punctuation. Table 2 gives an overview of the corpora used in the experiments.

For this study, the thematic range of TGermaCorp, which is composed of literary texts in standard German, was extended to include language of science, from a diachronic perspective. Extracts were selected from two texts from the second half of the nineteenth century: Friedrich Nietzsche's *Der Antichrist* (1894), representing humanities, and Gregor Mendel's *Versuche über Pflanzenhybriden* (1866), representing the natural sciences. Nietzsche's text was obtained from the *Digitale*

Kritische Gesamtausgabe Werke und Briefe.[5] Mendel's text was obtained from *Project Gutenberg*.[6] The token count for *Der Antichrist* is 30,652 and for *Versuche über Pflanzenhybriden* it is 20,129.

Each text was divided into 12 equal chunks which were paired to form 12 mixed files for annotation. Each annotator had to annotate and occasionally correct the tokenization of a mixed file of 4,831 tokens. Eight annotators performed (coarse-grained) POS tagging and lemmatization, following STTS annotation guidelines (Schiller *et al.* 1999) – see Lücking *et al.* (2016) for further details.

Before annotating the TGermaCorp extension, annotators underwent a five-week training period, supervised by two linguistically trained experts. The objective of the training period was to familiarize annotators with the tagset, so as to achieve consistent annotation. The first part of the training period focused on the rules and labels for lemmatization, using the annotation manuals and comparing results with gold-standard annotations (one week). During the second part of the training period (four weeks), the annotators had to complete two annotation tasks. Their results were inspected on a sample basis by the two supervisors. The findings as well as any examples worthy of discussion encountered by the annotators, were discussed in weekly meetings, in order to clarify annotation rules or agree on conventions in cases not unequivocally covered by the manuals. Annotators who successfully completed the training period then annotated the TGermaCorp extension.[7]

In order to test annotator self-agreement, 300 tokens from each text snippet were extracted and added at the end of the annotation file, so that 600 tokens were annotated twice by each annotator. Intra-rater agreement was calculated on these two annotations for each annotator by means of Cohen's Kappa (Cohen 1960). The results are presented in Table 3. The perfect agreement for ann7 is due to her recognizing the double annotation task and reconstructing her previous choices.

An inter-rater agreement study was also carried out. Four annotators annotated a set of 100 tokens from Nietzsche's *Antichrist*, while the other four annotators annotated a set of 100 tokens from Mendel's

[5] Provided at `http://www.nietzschesource.org` (CC BY-NC 3.0).

[6] `http://www.gutenberg.org/cache/epub/40854/pg40854.txt`

[7] Although twelve annotators were trained, only eight successfully completed the training period.

Annotator	Lemma	POS
annotator 1	0.98	0.98
annotator 2	0.99	0.99
annotator 3	0.98	0.96
annotator 4	0.99	0.99
annotator 5	0.98	0.97
annotator 6	0.96	0.94
annotator 7	1.00	1.00
annotator 8	0.97	0.92

Table 3:
Intra-rater agreement: summary

Level	All	ann1-gs	ann2-gs	ann3-gs	ann4-gs
lemma	0.92	0.91	0.94	0.97	0.88
POS	0.85	0.96	0.94	0.88	0.85

Table 4:
Inter-rater agreement: summary for the Nietzsche extract

Level	All	ann5-gs	ann6-gs	ann7-gs	ann8-gs
lemma	0.95	0.95	0.85	0.94	0.95
POS	0.91	0.90	0.89	0.89	0.90

Table 5:
Inter-rater agreement: summary for the Mendel extract

Versuche über Pflanzenhybriden. These two sets of tokens had previously been annotated and discussed by two linguistically trained annotators, thus providing a gold standard for comparison. Agreement values for the Nietzsche extract are given in Table 4, and those for the Mendel extract are given in Table 5. Column 'all' presents agreement values among all four annotators in terms of Fleiss' generalized Kappa (Fleiss 1971). The remaining columns provide each annotator's compliance with the Gold Standard (Cohen's Kappa). With all agreement scores exceeding the threshold of 0.81, annotations can be regarded as 'sound' (Krippendorff 1980) or 'almost perfect' (Rietveld and van Hout 1993).

As self-consistency (intra-rater) is even higher than mutual consistency (inter-rater) – although both sets of values are satisfactory – and as more annotators were involved in annotating the extension than the original corpus, slightly more diverse annotation values are expected for the extension. We therefore expect a slight decrease in tagger performance for the newly added snippets.

5.2 *Tagsets*

The choice of a tagset depends mainly on the language to be annotated. For German NLP, the Stuttgart-Tübingen TagSet (STTS) is widely recommended. It is used in the Tiger corpus and has been adopted for the annotation of TGermaCorp. Consequently, STTS is used in this work to train and evaluate POS tagging. However, there are some cases in linguistic studies where for a certain type of word, such as a verb, it is not relevant to differentiate between sub-groups, such as finite or non-finite. Another argument in favor of simplification is that training effort for annotators will be reduced. Instead of simply training our taggers for STTS and grouping results into a more coarse-grained tagset, we also investigated the potential accuracy of a simplified tagset. We therefore performed experiments based not only on the STTS but also on the simplified tagset, sSTTS.

Our experiments on Latin texts are primarily based on the Capitularies corpus, which was tagged using the *Computational Historical Semantics TagSet* (CHSTS).[8] Compared to STTS, the CHSTS is coarse grained, and does not distinguish between different types of verbs, adjectives, or pronouns. It does, however, distinguish nouns from named persons and named entities. We therefore expect further simplification not to have as great an impact on the accuracy of POS tagging as that observed with the simplified STTS. Consequently, we also used a simplified tagset sCHSTS in our experiments, which maps nouns (NN), persons (NP), and named entities (NE) to one single POS tag (N). Table 6 maps simplified equivalents to STTS and CHSTS.

The tagsets discussed so far encode morphological information to some extent (e.g. VVFIN vs. VVIMP) but do not explicate morphological categories as a whole. We therefore differentiate between *coarse-grained* and *fine-grained* (morphological) POS tagging in our experiments. Table 7 provides an overview of morphological tags. Categories or tags that are unique either to Latin ($\cdot^{la}$) or German ($\cdot^{de}$) corpora are marked accordingly. The Tiger corpus, the Capitularies and the Proiel corpus provide annotations of `case`, (comparison) `degree`, `gender`, `mood`, `number`, `person` and `tense`. In addition, the Latin texts are an-

[8] This tagset was developed within the framework of the Computational Historical Semantics project (`http://www.comphistsem.org/`) (Jussen *et al.* 2007).

Table 6: Mapping simplified tagsets for STTS and CHSTS

	sSTTS	STTS	sCHSTS	CHSTS
adjective	ADJ	ADJA, ADJD	ADJ	ADJ
adposition	AP	APPO, APPR, APPRART, APZR	AP	AP
adverb	ADV	ADV, PAV	ADV	ADV
article	ART	ART		
card. num.	CARD	CARD	NUM	NUM
conjunction	KON	KOKOM, KON, KOUI, KOUS	CON	CON
dist. num.			DIST	DIST
foreign	FM	FM	FM	FM
interjection	ITJ	ITJ	ITJ	ITJ
non-word	XY	XY	XY	XY
noun	N	NN, NE	N	NN, NE, NP
particle	PTK	PTKZU, PTKNEG, PTKVZ, PTKA, PTKANT	PTC	PTC
pronoun	P	PDAT, PDS, PIAT, PIS, PPER, PPOSAT, PPOSS, PRELAT, PRELS, PRF, PWAT, PWAV, PWS, PIDAT	PRO	PRO
truncation	TRUNC	TRUNC		
verb	V	VAFIN, VAIMP, VAINF, VAPP, VMFIN, VMINF, VMPP, VVFIN, VVIMP, VVINF, VVIZU, VVPP	V	V
pun. term.	$.	$.	$.	$.
comma	$,	$,	$,	$,
other pun.	$(	$(	$(	$(

Category	Tags
case	$*^{de}$, ablative^{la}, accusative, dative, genitive, locative^{la}, nominative, vocative^{la}
degree	$*^{de}$, comparative, positive, superlative
gender	$*^{de}$, feminine, masculine, neuter
mood	gerund^{la}, gerundive^{la}, imperative, indicative, infinitive^{la}, participle^{la}, subjunctive, supine^{la}
number	$*^{de}$, plural, singular
person	1, 2, 3
tense	future^{la}, future perfect^{la}, imperfect^{la}, past^{de}, perfect^{la}, pluperfect^{la}, present
voice^{la}	active^{la}, passive^{la}

Table 7: Morphological tags used for fine-grained POS tagging. The $\cdot^{la}$ tags are only used for Latin texts, whereas $\cdot^{de}$ tags are only used for the German Tiger corpus

notated with either active or passive `voice`. Note that tags common to both corpora can be mapped directly, even if they are not identical in appearance (e.g. “acc” in Tiger and “accusative” in the Latin texts). The morphological annotation of the two Latin corpora is much richer than that of the Tiger corpus.

5.3 *Lexicons*

For German, we extracted a lexicon from the German Wiktionary.[9] Extracting lemmas and syntactic words (including all grammatical categories available) from a Wiktionary instance in a thorough *and* robust way is not a trivial task. Even though guidelines and templates exist, they differ significantly between Wiktionary instances, and also vary in the way they are used within the same language. Our approach parses the HTML code of a Wiktionary instance that has been setup on a local server using the XML-dump from 2015-09-01.[10] This is, in our experience, more accurate than trying to parse MediaWiki sources directly, and it saves bandwidth on the official Wiktionary servers. We also used GermaNet version 11.0 (Hamp and Feldweg 1997; Henrich and Hinrichs 2010) as a lexical resource. In both cases, we are limited to the simplified STTS tagset, since the lexicons extracted to not have sufficient information to differentiate between different types of verbs, as STTS does. Future work may however include an STTS-compliant extraction and mapping of Wiktionary.

For Latin, we made use of the Frankfurt Latin Lexicon (FLL) (Mehler *et al.* 2015). As an equivalent to GermaNet, we included the lexical resources of a Latin WordNet (Minozzi 2008), available under an Attribution-ShareAlike 4.0 license.

For MarMoT, we used information about word forms and their POS. Table 8 gives an overview of the lexical resources used in the experiment.

5.4 *Embeddings*

Besides lexicons, word embeddings can be used as an additional resource to train specific taggers like MarMoT. In order to achieve reliable representations of word forms in a vector space, large corpora

[9] `http://de.wiktionary.org`

[10] `https://dumps.wikimedia.org/dewiktionary/20150901/`

Lexicon	Language	Tagset	Word forms
DE-Wiktionary	German	sSTTS	381,296
GermaNet	German	sSTTS	126,392
GermanAll	German	sSTTS	463,912
FLL	Latin	CHSTS	3,635,245
FLL	Latin	sCHSTS	3,631,179
LatinWordNet	Latin	sCHSTS	9,124
LatinAll	Latin	sCHSTS	3,631,199

Table 8:
Statistics of lexica used in the experiments

are required. We considered three corpora: The Gutenberg-DE Edition 13, the German Wikipedia, and the German newspaper "Süddeutsche Zeitung" (SZ). All corpora were tokenized using the PTBTokenizer contained in StanfordCoreNLP (Manning *et al.* 2014), lemmatized and tagged using MarMoT, and dependency parsed using Mate (Bohnet and Nivre 2012). All these tools and the corresponding processing pipeline are available (also as web-services) via the TextImager system (Hemati *et al.* 2016).

The Gutenberg-DE Edition 13 is a collection of classical German literature ranging from modern works back to a poem by Walther von der Vogelweide written in 1198. In contrast, the German Wikipedia corpus covers articles of the online encyclopedia which were extracted from a dump dating from 2016-02-03. The articles were parsed using Sweble (Dohrn and Riehle 2011) and converted into TEI P5. Finally, the newspaper corpus covers 23 volumes of the "Süddeutsche Zeitung" between 1992 and 2014. In our experiments, we use each corpus separately and all corpora combined (German-All).

For Latin, we use texts ranging from the 2nd to the 14th century. The documents stem from the Patrologia Latina (PL) corpus, the Monumenta Germaniae Historica, and the Central European Medieval Texts Series.[11] [12] [13] Most of the texts are from the 9th to the 12th century and were written by clerics. Since the Patrologia Latina does not contain annotations of dependency structures, it is not suitable for all the word embedding tools examined in this contribution. In order to explore the effect of incorporating dependency structure information

[11] `http://patristica.net/latina`

[12] `http://www.mgh.de/dmgh`

[13] `http://www.ceupress.com/books/html/CentralEuropeanMedievalTexts.htm`

into word embedding nonetheless, we included the Index Thomisticus. This corpus contains texts by Thomas Aquinas (1225–1274 AD). Its size is but a fraction of the Patrologia Latina, but it is annotated with dependency information (Passarotti 2015).

Table 9 gives an outline of the corpus statistics.

Table 9: Statistics of supplementary corpora used in the experiments

Corpus	Language	Sentences	Tokens
Gutenberg	German	24,766,958	440,896,599
Wikipedia	German	85,027,606	1,158,005,656
SZ	German	42,426,628	725,868,505
German-All	German	152,221,192	2,324,770,760
PL	Latin	5,598,592	133,158,974
Index Thomisticus	Latin	21,931	371,824

In order to compute word embeddings, we incorporate five different variants: The Mikolov model (Mikolov *et al.* 2013) optimizes word embeddings such that they can predict other context words occurring in a defined window. The model considers target-context word pairs inside a window of words to the right and to the left of the target word. Among other options, the text model chosen can be either the *continuous bag of words model* (cbow) or the *skip-gram model* (skip). The effect of varying this parameter is examined in the experiments discussed in Section 6.1.2. FastText (Bojanowski *et al.* 2017) is a library and tool to learn word embeddings as well as sentence classifications. Pennington *et al.* (2014) developed GloVe, which we also examined as an alternative to learn word embeddings. Levy and Goldberg (2014) modified the skip-gram model of Mikolov *et al.* (2013). They used dependency contexts instead of a window-based word context. Komninos and Manandhar (2016) introduced a variant of the skip-gram model that combines Mikolov skip-grams with those derived from dependency trees. Each target word optimizes word embeddings such that maximum probabilities of other words within distance one and two in the dependency tree are calculated. A weighting according to distance is applied. Words with distance one from the target word are counted twice. These word-word predictions behave similar to the window model of Mikolov *et al.* (2013). Dependency

parses are also used to filter coincidental co-occurrences (Komninos and Manandhar 2016).

5.5 *Morphological analyzers*

This article focuses on standard tools for natural language processing that are (i) generic, in the sense that they can in principle be applied to any language and genre, and (ii) produce an unambiguous annotation. However, there are resources and tools developed for a specific language that provide detailed information on lexical units and are widely used in the community.

Morphological analyzers provide information about inflected forms regarding morphology and lemmatization. They may also include information regarding segmentation (e.g. Lemlat). All of the following analyzers perform an out-of-context analysis, that is, they process each inflected form individually and may return multiple results. Furthermore, these results do not necessarily cover the actual valid result (when context is considered). Thus we cannot directly compare these resources to the tagging tools analyzed in our experiments. However, we can perform a coverage analysis in order to get an overall impression of the potential of the analyzers. For this purpose, we discard numerals and punctuation. Table 10 summarizes the coverages of the following morphological analyzers. Morpheus is a web service as part of the Perseus project (Crane 1991), which provides morphological analyses for Greek and Latin.[14] Lemlat 3.0 (Passarotti *et al.* 2017) is a morphological analyzer based on a lexical database. LatMor (Springmann *et al.* 2016) is a finite-state morphology for Latin, which, on our corpora, reached the best coverage.

Analyzer	Capitularies	Proiel
LatMor	99.527	99.426
Lemlat	97.616	99.495
Morpheus	92.371	96.297

Table 10:
Coverage of morphological analyzers on Latin corpora in percent

[14] To evaluate the coverage of Morpheus, we used a Python project (`https://github.com/tmallon/morpheus`) by Timothy Mallon, which wraps and caches requests on the Morpheus web service.

6 EXPERIMENTS

In this section, we evaluate the performance of various NLP tools. We do not seek to find only one perfect combination of tool and resources in order to proclaim a winner. To attempt to do so would require extensive hyperparameter experiments and tuning of each tool. Our experiments showed that this is almost impossible since some tools take hours to train a model on our compiled training/test partitions *and* have a wide parameter space to explore. And even then the results would be valid for the examined corpora but might differ significantly for other corpora. This is because the process of optimizing parameters to maximize accuracy for a given data set might lead to over-fitting for that specific task.

So our focus is rather on the practitioner: What accuracy can one generally expect from a given NLP approach – used off-the-shelf and using default hyperparameters – and how much time do you need to invest for training and predicting? Furthermore, we are interested in how significant the differences are between older and more modern approaches. Does “old” automatically need to stand for “no longer relevant”?

We start by examining POS and fine-grained POS tagging, followed by lemmatization. Finally, we evaluate what level of accuracy we can expect when putting the pieces together.

6.1 *Tagging*

Table 11 shows POS tagging accuracy, achieved without using any optimization or adding any supplementary resources for training. LemmaTag performed best on almost all German corpora, followed by MarMoT and FLORS. The picture becomes more diverse when considering the results for the Latin texts. While LemmaTag still performs best on the Capitularies, accuracy drops significantly for Proiel as well as (to a minor degree) for the two OD scenarios. On average, MarMoT and the tagging veteran TreeTagger are only 1.40 % apart. Another outlier worth noticing is FLORS, trained on Tiger and tested on TGermaCorp, which is 0.47 % above LemmaTag (second in place for this setting), followed by Mate and Lapos. As expected, we observed a significant drop in accuracy for OD experiments, compared to ID experiments. For example, MarMoT achieves 98.02 % accuracy on the

Table 11: POS accuracy for STTS and CHSTS in %. Lines written in italics indicate tagging results of out-of-vocabulary tokens only

	FLORS	Lapos	LemmaTag	MarMoT	Mate	OpenNLP	RDR	Stanford	TnT	TreeTagger
TG	92.33	93.12	**93.44**	93.34	92.89	91.64	91.09	91.67	91.63	92.22
TG	*82.07*	*84.69*	***86.02***	*86.01*	*84.54*	*80.73*	*69.41*	*76.05*	*81.30*	*79.85*
Tiger	97.69	97.84	**98.58**	98.02	97.88	96.84	96.72	97.17	97.23	97.09
Tiger	*91.59*	*93.00*	***93.93***	*93.74*	*93.05*	*90.18*	*81.20*	*85.99*	*90.78*	*88.49*
TG→Tiger	89.70	85.69	**90.69**	90.59	88.94	87.09	85.59	85.62	88.41	88.33
TG→Tiger	*80.57*	***80.77***	*79.95*	*80.61*	*75.34*	*72.09*	*64.13*	*65.65*	*74.08*	*71.73*
Tiger→TG	**90.59**	89.19	90.12	89.10	89.22	87.75	86.70	87.52	88.52	88.26
Tiger→TG	***78.44***	*69.40*	*72.42*	*68.83*	*69.74*	*66.37*	*57.44*	*60.07*	*65.70*	*64.95*
Capit	95.44	96.08	**96.18**	96.10	95.79	94.83	95.43	94.83	95.47	95.17
Capit	*78.72*	***83.54***	*80.28*	*83.22*	*81.79*	*77.04*	*72.90*	*62.95*	*78.30*	*74.95*
Proiel	93.11	**95.62**	92.00	95.49	95.56	93.51	94.11	92.96	94.36	94.32
Proiel	*79.84*	*84.83*	*76.76*	***85.10***	*83.82*	*80.13*	*73.32*	*66.03*	*76.74*	*76.64*
Capit→Proiel	84.72	**87.73**	86.20	87.58	86.80	82.62	84.66	82.12	85.79	85.50
Capit→Proiel	*70.65*	*77.32*	*70.91*	***77.72***	*75.41*	*65.22*	*63.85*	*51.91*	*69.79*	*68.33*
Proiel→Capit	62.44	63.48	61.54	63.05	**63.85**	62.31	61.42	54.95	62.94	61.22
Proiel→Capit	*42.79*	*43.39*	*42.01*	*44.00*	***44.56***	*42.42*	*39.51*	*28.43*	*42.08*	*38.60*
average	88.25	88.59	88.60	**89.16**	88.87	87.07	86.97	85.86	88.04	87.76
average	*75.59*	*77.12*	*75.28*	***77.41***	*76.03*	*71.77*	*65.22*	*62.13*	*72.35*	*70.44*

Tiger corpus (ID). But when Tiger is used as the training corpus to tag TGermaCorp, results drop by about 9%. The problem becomes even more apparent when a relatively small corpus such as Proiel is used to tag a larger corpus like the Capitularies. In this case, accuracy drops from 95.62 % (Proiel ID) to 63.48 % (Proiel→Capitularies) when using Lapos. However, this is partly because Proiel does not contain punctuation, so a model trained on that corpus is bound to produce more errors when applied to the Capitularies. Since a practitioner is often limited to using pre-trained models, OD experiments reveal a much more realistic view of what can be expected. In Section 6.1.2, we will examine how and to what degree accuracies in OD scenarios can be improved by supplementary resources. Our results show that given a specific tagger, accuracy may vary heavily across different resources. We also observe that the taggers perform quite differently for the same resource. This finding suggests, that it may be worth the effort to try more than one tagger on a given POS tagging task.

Taggers may vary considerably in the way they can cope with words which have not been part of the training set (out-of-vocabulary items). Table 11 lists the accuracy for out-of-vocabulary items in italics. Figure 3 shows a visualization of the differences compared to all tokens as heatmap. The more saturated the cell, the higher the delta. The maximum delta of 31.88% is reached by Stanford tagger on the Capitularies. Lapos and MarMoT perform best from the perspective of how well they can cope with out-of-vocabulary items. In contrast Stanford tagger, RDR and TreeTagger mark the other end of the spectrum. Figure 3 also reveals that the delta for Tiger→TG is much higher than for TG→Tiger which is in contrast to Capit→Proiel vs. Proiel→Capit: In the German out-of-domain scenario accuracy drops significantly more when Tiger, being the much larger corpus, is used to tag TGermaCorp. For Latin it is the other way around: Accuracy drops much more for Proiel being used as training set to tag the significantly larger Capitularies corpus.

What are the main reasons for tagging errors? Figure 4 depicts in a bipartite graph how often POS-tagging errors have occurred, using LemmaTag on Tiger. The arrows from the gold-standard (top) to the test results (bottom) represent the range of variation in POS-tagging errors, and their relative frequency of occurrence. For better readibility, the figure only shows the ten most frequent nodes for gold

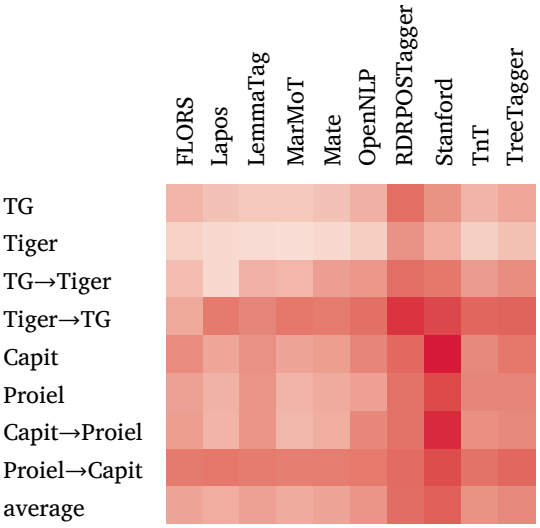

Figure 3: Heatmap depicting the degree to which accuracy drops when only out-of-vocabulary tokens are considered. The more saturated the cell color, the higher the delta

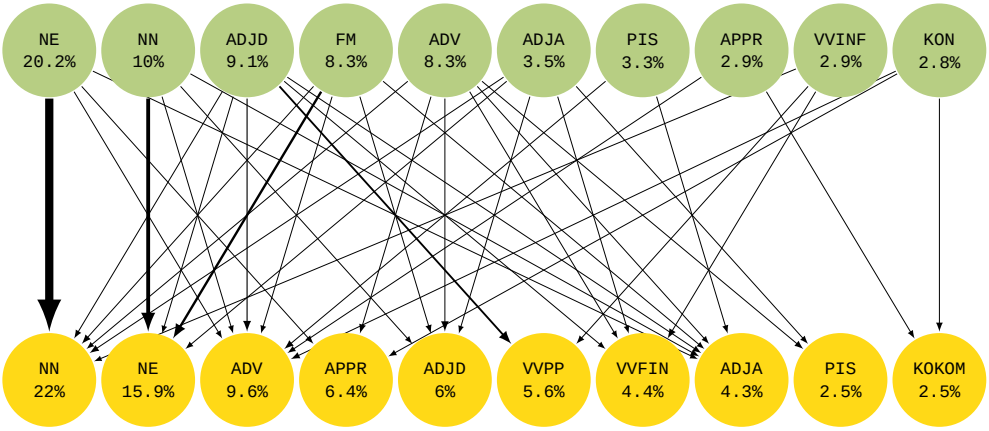

Figure 4: Depiction of the distribution and extent of erroneous POS tagging, comparing gold-standard tags (top) with test-results (bottom), using STTS for LemmaTag on Tiger

standard, and for test results, while arrows indicate at least ten mismatches. Percentages represent the ratio of all incoming or outgoing arrows with respect to the overall total. Apparently, most errors are caused by named entities recognized as nouns and vice versa. Like-

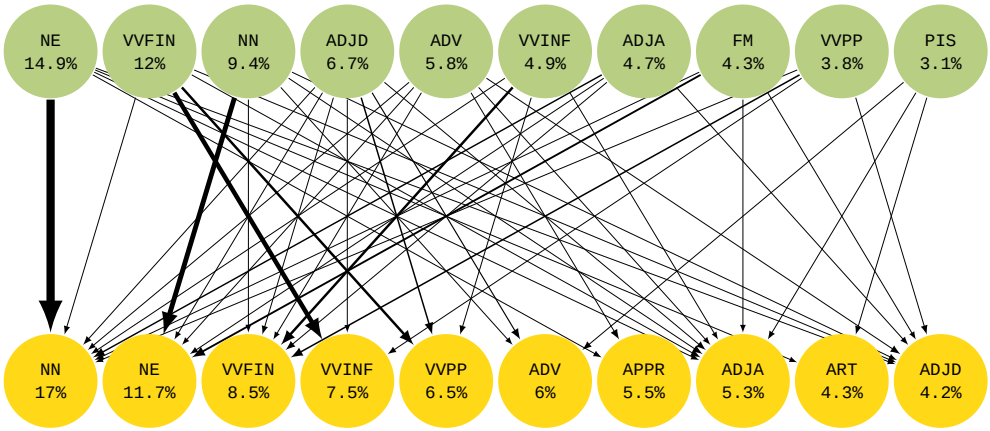

Figure 5: Depiction of the distribution and extent of erroneous POS tagging, comparing gold-standard tags (top) with test-results (bottom), using STTS for all taggers on Tiger

wise, finite verbs are often erroneously tagged as non-finite verbs and vice versa. Are these observations specific to LemmaTag or do other taggers show similar behavior? Compare Figure 4 with Figure 5, which shows the sum of all mismatches over all taggers combined. Percentages in the upper row code the fraction of errors where the corresponding POS was tagged erroneously; percentages in the lower line code the fraction of errors where this target POS was chosen. The figure is based on the same filter criteria. As before discriminating named entities and nouns as well as finite and non-finite verbs are the main sources of error.

The POS-tagging accuracy results shown in Table 11 for ID experiments are the average of a three-fold cross-validation. The choice of how to partition a corpus into a training set and a test set may have a significant impact, and thus make comparision between POS tagging results from the literature more difficult. The distribution of the deltas between the minimum and maximum value encountered for POS tagging in cross-validations (Figure 6) clearly shows that delta values depend on the size of the corpus. The smaller the corpus to be partitioned, the higher the probability that the training set will fail to cover words or patterns that are part of the test set.

In Section 5.2, we introduced a mapping from STTS to a simplified version that abstracts from variants of verbs, pronouns, etc.

	FLORS	Lapos	LemmaTag	MarMoT	Mate	OpenNLP	RDRPOSTagger	Stanford	TnT	TreeTagger
TG	0.36	0.55	0.16	0.47	0.41	0.19	0.06	0.22	0.63	0.42
Tiger	0.09	0.01	0.08	0.03	0.08	0.08	0.02	0.03	0.10	0.12
Capit	0.30	0.26	0.17	0.28	0.27	0.28	0.10	0.36	0.45	0.33
Proiel	1.07	0.75	0.54	0.35	0.39	1.19	0.81	0.77	1.03	0.57

Figure 6: Distribution of deltas $acc_{max} - acc_{min}$ for POS-taggers in %, based on a three-fold cross-validation. The more saturated the cell color, the higher the delta

The Latin tagset CHSTS is similar to sSTTS already, but still discriminates nouns from named entities and named persons. Consequently, we also examined the impact of an abstraction of the Latin tagset, sCHSTS. As some use cases in computational linguistics do not require an explicit distinction for verbs, pronouns, and other POS, the question is whether this abstraction results in significantly higher accuracy. Table 12 contrasts the STTS/CHSTS-based POS results for Tiger and the Capitularies with the simplified tagsets, sSTTS and sCHSTS.[15] All taggers show improvement after the abstraction. On average Tiger gains 1.09% while accuracy for Capitularies is increased by 0.32%. Since sCHSTS only abstracts nouns, the improvement is understandably less.

We already noted that mismatches between named entities and nouns, as well as variants of verbs, are a major source of POS tagging errors. Shifting to a simplified tagset brings into focus other errors that were already present in the original tagsets. Figure 7 shows the percentage of erroneous POS using sSTTS for all taggers on Tiger. As before, the figure shows only the ten most frequent types for gold standard and for test results, with arrows to indicate at least ten mis-

[15] For two out of three train/test partitions of Tiger, Lapos frequently tagged commas not as "$," but as "$(", "V" and other forms. As this error was not observed for other corpora nor other taggers, we presumed that this was an error in Lapos and therefore explicitly set the correct tag for commas in this specific case.

Table 12:
POS accuracy for all tagsets on Tiger and Capitularies in %

	Tiger STTS	Tiger sSTTS	Capit CHSTS	Capit sCHSTS
FLORS	97.69	**98.74**	95.44	**95.65**
Lapos	97.84	**98.69**	96.08	**96.31**
LemmaTag	98.58	**99.18**	96.18	**96.51**
MarMoT	98.02	**98.88**	96.10	**96.35**
Mate	97.88	**98.90**	95.79	**96.09**
OpenNLP	96.84	**98.11**	94.83	**95.18**
RDRPOSTagger	96.72	**97.92**	95.43	**95.88**
Stanford	97.17	**98.37**	94.83	**95.18**
TnT	97.23	**97.93**	95.47	**95.74**
TreeTagger	97.09	**97.86**	95.17	**95.60**
average	97.51	98.46	95.53	95.85

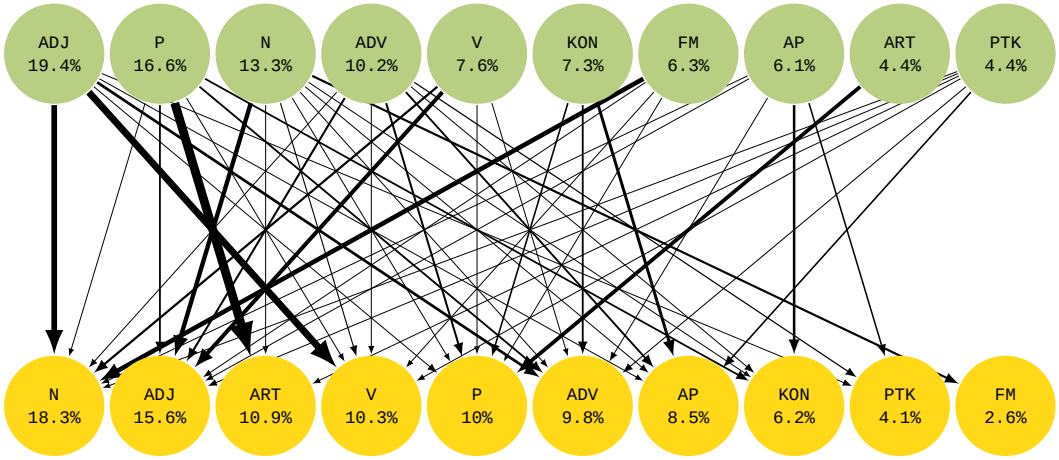

Figure 7: Percentage of POS tagging errors using sSTTS for all taggers on Tiger

matches. The most dominant errors are adjectives erroneously tagged as nouns or verbs. Another frequent error stems from pronouns beings tagged as articles. These errors were present in STTS as well but were dominated there by the much more frequent problem of distinguishing nouns from verbs.

So far we have only considered coarse-grained POS tagging. Now, we shed light on what accuracy can be achieved for fine-grained POS tagging. Instead of lining up result tables as we did for POS, we propose a graphical representation to depict tagging accuracy for the

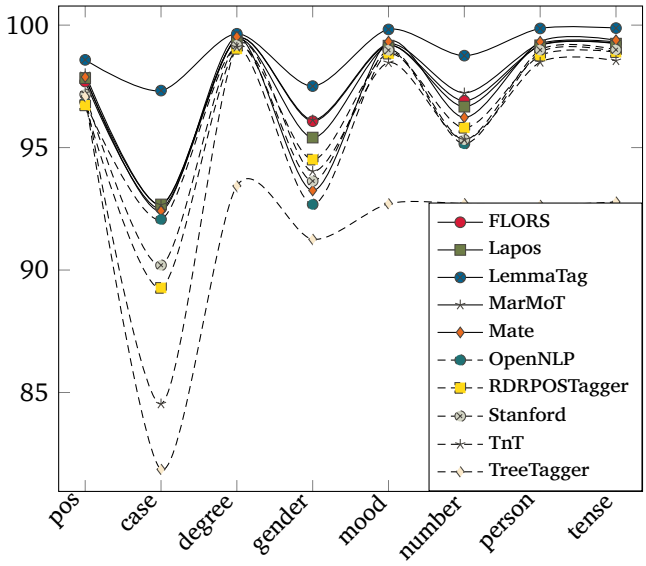

Figure 8:
Fine-grained POS tagging of the Tiger corpus (accuracy in %)

Tiger corpus (Figure 8) and for the Capitularies (Figure 9). For both corpora, results are relatively good for `degree`, `mood`, `person`, and `tense`, whereas `case`, `gender`, and `number` appear to be more challenging. All tools show similar behavior with respect to the accuracy for a given morphological category. LemmaTag performs exceptionally well on all categories, while TreeTagger marks the lower end of the spectrum.

6.1.1 Majority voting

When more than two annotations are generated for the same task, a majority-vote approach can be applied. Assuming that the majority is more likely to be right when tagging a specific token, erroneous outliers can be compensated for, thus leading to better results. We distinguish three groups of taggers: By "top3" and "top5" we denote the three or five most recently published (see Section 4): LemmaTag, FLORS, RDRPOSTagger, MarMoT, and Mate. By "all" we denote the ten taggers examined in our study. We also rely on (descending) order of publication of the taggers to resolve tie situations in majority votes.

As Table 13 shows, applying a majority vote improves performance in most cases. Unexpectedly, using the three most recently published taggers (LemmaTag, FLORS, and RDRPOSTagger, according to

Figure 9:
Fine-grained POS tagging of the Capitularies (accuracy in %)

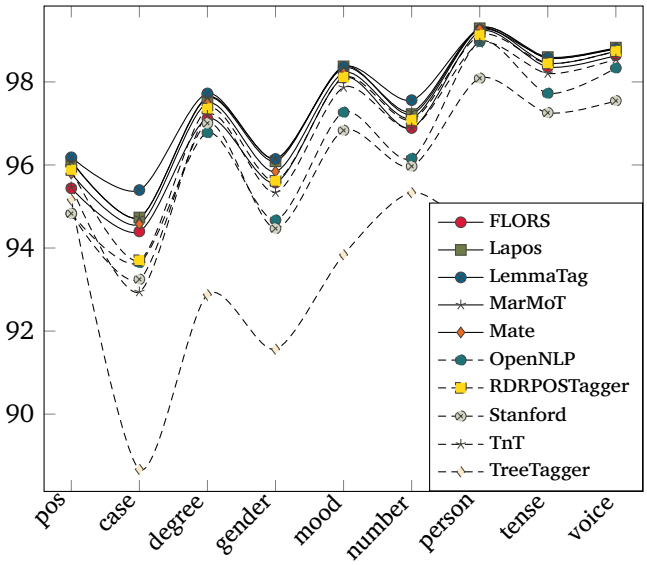

Table 13:
Majority vote tagging of POS (STTS/CHSTS) (accuracy in %)

Corpora	Best	Top3	Top5	All
TG	93.44	93.63	**93.91**	93.86
Tiger	**98.58**	98.30	98.37	98.29
TG→Tiger	90.69	90.47	91.37	**91.39**
Tiger→TG	**90.59**	90.41	90.44	90.08
Capit	96.18	96.20	**96.29**	96.28
Proiel	95.62	94.70	95.78	**96.04**
Capit→Proiel	87.73	86.49	87.82	**87.92**
Proiel→Capit	63.85	63.51	64.61	**64.76**

our list) cannot be recommended, as it improves accuracy in only two out of eight cases. However, by adding MarMoT and Mate to this list, accuracy exceeds the baseline in six out of eight cases. A similar improvement is observed when using all taggers.

We then went a step further and evaluated the performance of majority votes for any combination of taggers, in any possible order (to solve ties). With 10 taggers available, 9,864,000 combinations were computed. Table 14 lists the best combinations for fine-grained (STTS) POS tagging of the Tiger corpus, using a fixed number of taggers. In this scenario, the best result (98.44%) is achieved when using LemmaTag, Mate, TnT, and FLORS (in that order), which is 0.14% above

#	Acc.	Taggers (ordered chronologically, most recent first)
3	98.39	LemmaTag, FLORS, Mate
4	**98.44**	LemmaTag, Mate, TnT, FLORS
5	98.37	LemmaTag, RDRPOSTagger, FLORS, MarMoT, Mate
6	98.41	LemmaTag, RDRPOSTagger, Mate, FLORS, MarMoT, TreeTagger
7	98.33	LemmaTag, RDRPOSTagger, Mate, TnT, MarMoT, Lapos, FLORS
8	98.36	LemmaTag, RDRPOSTagger, Mate, Lapos, MarMoT, TreeTagger, TnT, FLORS
9	98.29	LemmaTag, Mate, MarMoT, TnT, TreeTagger, FLORS, Stanford, RDRPOSTagger, Lapos
10	98.31	LemmaTag, Mate, FLORS, RDRPOSTagger, Stanford, MarMoT, TnT, Lapos, TreeTagger, OpenNLP

Table 14: Best results for majority vote POS-tagging (STTS) of Tiger by number of taggers (accuracy in %)

the best single performer (LemmaTag). Our results suggest that, in practice, the relatively small gain in performance does not justify the great effort for training the individual tools in order to use majority vote tagging.

6.1.2 Supplementary resources

The first experiment presented in this article examined the performance of various tools on POS tagging, with no hyperparameter tuning and no additional resources whatsoever. We observed that, even though all tools produced reasonable results for ID scenarios, switching to OD settings had a severe impact on accuracy: The training data cannot cover the morphological and grammatical diversity, or the specific characteristics of the test domain. This problem becomes even more severe when only small training corpora are available. Creating well-annotated corpora as gold standard for tagging is a tedious and complex task. How can additional resources that are easier to produce be used to augment models, and what impact can be expected on the accuracy of our scenarios?

Lexical resources as well as word embeddings can be used with MarMoT to support the training of POS tags, while FLORS can benefit from additional unstructured texts. Because of the time required to train FLORS models, we focused on MarMoT to examine the impact of different resources on overall performance. We started by examining POS tagging of Latin corpora, using different combinations of lexicons

Table 15: POS (CHSTS) accuracy in % using MarMoT, for different types of word embedding, based on two Latin corpora

Emb. corp.	Corpora	No em.	Mik.	F.Text	GloVe	Kom.	Levy
PL	Capit	96.10	**96.26**	96.23	96.01	–	–
Thom.	Capit	**96.10**	96.08	96.09	96.06	96.03	96.01
PL	Pro	95.49	**96.85**	96.75	95.90	–	–
Thom.	Pro	95.49	95.67	95.56	95.75	**95.83**	95.78
PL	Cp→Pro	87.58	88.68	**88.69**	86.75	–	–
Thom.	Cp→Pro	**87.58**	87.28	87.35	86.40	86.33	86.14
PL	Pro→Cp	63.05	**69.61**	68.77	64.91	–	–
Thom.	Pro→Cp	63.05	63.65	63.47	62.92	**64.25**	64.24

and word embeddings. The word-embedding approaches of Komninos and Levy can only be applied to the Index Thomisticus, because the Patrologia Latina corpus lacks dependency information.

We used default values for Mikolov, FastText, GloVe, Komninos, and Levy, while aiming to keep results for the different approaches comparable. We therefore used a feature vector size of 100 and 5 iterations (Levy used 1 by default, which we changed to 5). As the default number of iterations for GloVe is 25, we did not limit the number of iterations to 5. For German, we set the minimum word frequency to 10 in order to compute embeddings within reasonable timeframe. For Latin, we used a minimum word frequency of 1.

Table 15 shows that using word embeddings based on the Patrologia Latina improved tagging results. Mikolov performs best in most cases, followed by FastText and GloVe. The most significant boost in terms of accuracy is achieved for the OD setting Pro→Cp. In this case, training POS tags based on the rather small Proiel corpus benefits best from the additional information provided by word embeddings. In two cases, GloVe has a negative impact compared to the baseline of using no word embeddings.

In contrast, the results of using Index Thomisticus as a basis for word embeddings are rather inconclusive. In about half the cases the baseline is not met. When Proiel is examined in-domain or out-of-domain, Komninos yields the best results, but in the other scenarios, it does not reach baseline accuracy either. Using dependency information may therefore have a positive impact, but a great deal depends on the corpora being processed. The Index Thomisticus consists only of texts from the 13th century, whereas the Capitularies date from

Corpora	Lexicon	Tagset	No em.	PL skip	PL cbow
Capit	none	CHSTS	96.10	96.26	95.02
Capit	FLL	CHSTS	96.52	**96.60**	95.52
Capit	none	sCHSTS	96.35	96.49	95.44
Capit	FLL	sCHSTS	96.67	**96.69**	95.77
Capit	La-WN	sCHSTS	96.37	96.49	95.45
Capit	all	sCHSTS	96.68	96.69	95.78
Cp→Pro	none	CHSTS	87.58	88.68	85.29
Cp→Pro	FLL	CHSTS	88.59	**88.79**	86.03
Pro→Cp	none	CHSTS	63.05	69.61	68.65
Pro→Cp	FLL	CHSTS	69.88	**72.82**	71.18

Table 16:
POS accuracy in %, using MarMoT, for combinations of lexicons and Mikolov-embeddings on Capitularies as well as OD-settings of Capitularies and Proiel

the Merovingian and Carolingian periods, while Proiel is based on ancient and early Christian Latin. Furthermore, the Index Thomisticus is rather small, at least compared to the Patrologina Latina. Finally, the Index Thomisticus treebank is completely set in lower-case, which may also contribute to the bad overall results.

Table 16 reveals that including lexical resources to train MarMoT consistently improves accuracy in ID as well as OD experiments. The most significant improvement can be observed when the relatively small Proiel corpus is used to tag the Capitularies. By including the Frankfurt Latin Lexicon (FLL), accuracy can be increased by 6.83% to 69.88%. Even a small lexicon like that extracted from the Latin WordNet, which is only a fraction the size of the FLL, has a positive impact on the results. As already noted, Proiel does not contain punctuation, and neither do the lexicons: thus the improvements result from the additional lexical resources, and are not merely a fix for a single shortcoming. Table 16 also shows that skip-gram clearly outperforms the continuous bag of words (cbow) model when using Mikolov. Finally, results confirm that using the simplified tagset sCHSTS over CHSTS also provides better results when supplementary resources are used. The best improvement can be achieved by incorporating skip-gram and FLL for the Proiel→Capit scenario, which raises accuracy from 63.05% to 72.82%.

In order to study the impact of supplementary resources on German, we rely on two lexica: one extracted from the German Wiktionary and the other from GermaNet (see Section 5.3). We used five

methods (Mikolov, FastText, GloVe, Komninos, and Levy) to compute word embeddings on four different corpora: Gutenberg, Süddeutsche Zeitung (SZ), Wikipedia, and a merged corpus thereof (see Section 5.4). Combined with four different ID and OD scenarios, for two different tagsets, there are too many options to be shown in detail. We rather focus on specific aspects, and then describe the whole.

We first examined the impact of different types of word embeddings, based on the four corpora available for this experiment. Table 17 shows the results for POS tagging, using MarMoT, on a selection of ID and OD settings for Tiger and TGermaCorp, using STTS. As we already saw for Latin, our findings indicate that using word embeddings can have a positive or negative effect on accuracy, depending on the method used and the parameters applied. Mikolov and Komninos yielded the best accuracy, even for OD scenarios. FastText was mostly on a par with Mikolov, whereas using GloVe gave slightly worse results, but was still better than the baseline of no embeddings at all. The expected accuracy gain when including dependency information for training with Komninos and Levy was much more obvious in the experiments on German in comparison with the Latin corpora. However, we assume that this result cannot be generalized to the language examined, but rather to the specific characteristics of the Latin corpus used.

Table 17: POS (STTS) accuracy in % using MarMoT, for different types of word embedding, based on four German corpora

Emb. corp.	Corpora	No em.	Mik.	F.Text	GloVe	Kom.	Levy
Gutenberg	Tiger	98.02	98.12	98.11	98.06	98.19	98.16
SZ	Tiger	98.02	98.20	98.20	98.11	98.29	98.22
Wikipedia	Tiger	98.02	98.17	98.16	98.05	98.25	98.19
Merged	Tiger	98.02	98.22	98.22	98.11	**98.32**	98.28
Gutenberg	Tig→TG	89.10	90.17	90.10	89.88	90.11	90.07
SZ	Tig→TG	89.10	89.81	89.67	89.92	89.67	89.54
Wikipedia	Tig→TG	89.10	89.79	89.72	89.84	90.07	89.92
Merged	Tig→TG	89.10	**90.45**	90.17	90.21	90.37	90.33
Gutenberg	TG→Tig	90.59	91.98	90.72	91.60	91.61	91.62
SZ	TG→Tig	90.59	92.51	92.18	91.19	93.10	92.84
Wikipedia	TG→Tig	90.59	92.37	92.64	91.54	93.43	93.44
Merged	TG→Tig	90.59	92.96	93.02	91.83	**93.80**	93.44

Corpora	Tagset	No em.	Mik.	F.Text	GloVe	Kom.	Levy
TG	STTS	93.34	94.03	94.05	93.64	**94.08**	93.79
Tiger	STTS	98.02	98.22	98.22	98.11	**98.32**	98.28
TG→Tiger	STTS	90.59	92.96	93.02	91.83	**93.80**	93.44
Tiger→TG	STTS	89.10	**90.45**	90.17	90.21	90.37	90.33
TG	sSTTS	96.02	**96.51**	96.43	96.27	96.47	96.40
Tiger	sSTTS	98.88	99.00	98.98	98.90	**99.04**	99.02
TG→Tiger	sSTTS	94.83	95.75	95.55	94.85	**96.15**	96.13
Tiger→TG	sSTTS	92.41	93.22	93.10	93.10	**93.35**	93.07

Table 18: POS accuracy in % using MarMoT, for different types of word embedding, based on all German corpora (merged)

Do we get a different picture when we consider the simplified version of STTS, which abstracts from different types of verbs, etc.? Table 18 presents the results for POS tagging, comparing STTS with sSTTS. Apart from the general improvement of accuracy by switching to sSTTS, we observe the same behavior as with plain STTS when embeddings are introduced. Using embeddings based on Komninos yields the best results. The accuracy for Tiger reaches the 99% mark.

In Section 4, we mentioned an approach for POS tagging based on Bidirectional Long Short-Term Memory Recurrent Neural Networks (BLSTMRNN; Wang *et al.* (2015)). Because of limited hardware resources and the extensive time needed to train the networks, we did not fully include it in our experiments. However, we did perform a POS tagging experiment on Tiger (ID), in order to estimate the performance of this class of taggers compared to others. We include it in the section discussing supplementary resources because our implementation strictly requires word embeddings (rather than using them as an option as MarMoT does). We used Mikolov and the merged German corpus for this task. Using BLSTMRNN achieves an accuracy of 98.29%, which is comparable to MarMoT (98.22%), using the same word embedding.

So far we have only considered word embeddings as a supplementary resource for German POS tagging. In the following, we examine the impact of the two lexicons based on Wiktionary and GermaNet. Table 19 shows the accuracy for combinations of lexicons and embeddings on the Tiger corpus. Since we observed that Levy was outperformed by Komninos based on the merged German corpus, we only considered one option (Komninos) as word embedding. Wik-

Table 19: POS (sSTTS) accuracy in % for combinations of German lexica and word embeddings, using MarMoT with Komninos, based on the merged German corpus

	None	GNet	Wikt.	AllLex	Emb	All&Emb
TG	96.02	96.11	**96.26**	96.24	96.47	96.46
Tiger	98.88	98.90	**98.95**	98.94	99.04	99.04
TG→Tig	94.83	94.96	95.14	**95.16**	96.15	96.16
Tig→TG	92.41	92.50	92.56	**92.61**	93.35	93.33

tionary and, to a lesser extent, GermaNet provided a slight gain in performance. Merging both resources (AllLex) had no significant effect. Combining lexical resources and embeddings did not produce the expected improvement in accuracy. In fact, our findings indicate that, at least in this setting, using both types of resources may even lead to a slight drop in accuracy. This scenario suggests that it is better to rely solely on embeddings rather than lexical resources or a combination of both.

We therefore conclude that including lexical resources for POS tagging is consistently beneficial. Incorporating word embeddings may improve results in many cases, especially in OD scenarios. However, depending on the corpus being processed, as well the method and parameters being used to compute the embeddings, the effect may also be negative, as seen in our Latin experiments. For Mikolov, the skip model outperforms the continuous bag of words model.

6.1.3 RDR-based POS tagger on external taggers

Our experiments have shown that the RDR-based POS tagging approach produced only modest results. So far, however, we have simply applied the out-of-the-box procedure, using the internal, lexicon-based tagger for initial tagging, as a practitioner will most likely do. As Nguyen *et al.* (2016) have demonstrated, using an external tagger can yield better results.[16] They report an accuracy of 96.28 using the internal tagger for initial tagging and 97.46 using TnT, which constitutes an improvement of 1.18. In our experiment, we measured 96.72 as the baseline when using the internal tagger. By integrating TnT as the initial tagger, we achieve an accuracy of 97.62, an improve-

[16] According to personal communication with the authors, the data published in Nguyen *et al.* (2016) is not based on 10-fold jack-knifing, which they nonetheless recommend to achieve better results. Following their recommendation, we applied 10-fold jack-knifing in our experiments.

ment of 0.9 compared to the baseline and 0.39 compared to TnT. Can this approach also improve the results for the MarMoT tagger, among the most successful taggers on the Tiger corpus in our experiments, with an accuracy of 98.02? When using MarMoT as the initial tagger for the RDR-based POS tagging approach, accuracy reaches 98.05, a slight improvement. This shows that the method can significantly improve the results of mediocre taggers, and may even have a positive impact on the best performing taggers. However this improvement massively increases computation costs in both training and testing.

6.1.4 Pipeline vs joint tagging

So far, we have examined each category of tagging independently. But as soon as morphological tagging is included, the question of maximum accuracy arises, when all tags have to fit the gold standard. We examined two different approaches: pipeline-learning and (what we call) "joint-learning". In pipeline-learning, we train and test each category independently. Then we combine all results to compute the overall accuracy. The advantage of this approach is that it can be performed by virtually any tagger. One of its major disadvantages, however, is that we cannot integrate any dependencies between categories into the learning process. Thus, it is theoretically possible for a noun to be tagged with a tense because each category is learned and tested independently.

A joint approach (in our sense) integrates all grammatical categories into one tag, which helps to capture dependencies. However, the tagset space becomes considerably larger: the Tiger corpus requires 53 STTS tags, but when the tags required to capture morphology are added, the total rises to 694 joint tags. Likewise, the Capitularies use 19 CHSTS tags, and 913 joint tags. In our experiments, taggers like Lapos and Stanford could not handle such a vast tagset space.

Table 20 shows accuracy results for joint and pipeline tagging on Tiger, Capitularies, and Proiel (ID/OD). As Figures 8 and 9 have already shown, accuracy varies considerably from category to category. While results for POS are relatively stable across all taggers, results for `case` and `gender` are significantly worse. The results achieved by pipeline learning are unlikely to be better than the individual results for each category.

Table 20: Accuracy in % for joint and pipeline tagging, on Tiger, Proiel, and Capitularies (STTS/CHSTS tagset)

	Tiger		Capitularies		Proiel		Capit→Proiel		Proiel→Capit		Average	
	Pipe	Joint	Pipe	Joint	Pipe	Joint	Pipe	Joint	Pipe	Joint	Pipe	Joint
FLORS	86.25	88.43	87.17	89.47	77.72	82.40	56.67	61.61	36.61	43.06	68.88	72.99
Lapos	85.99	–	88.74	–	83.02	–	61.43	–	41.19	–	**72.07**	–
LemmaTag	**93.72**	**93.68**	**89.58**	90.03	76.99	73.10	**62.11**	**63.43**	37.88	38.09	72.06	71.67
MarMoT	86.99	89.69	88.60	**90.70**	**83.03**	**84.37**	60.84	63.18	40.69	**43.25**	72.03	**74.24**
Mate	83.56	84.90	87.72	89.95	82.04	83.33	59.49	61.66	41.05	42.33	70.77	72.43
OpenNLP	81.88	85.54	84.40	87.57	75.33	79.01	51.04	56.06	36.68	38.61	65.87	69.36
RDRPOSTag.	82.16	85.97	87.83	89.43	82.13	83.02	60.46	61.72	**41.30**	40.87	70.78	72.20
Stanford	81.70	–	85.61	–	74.75	–	52.36	–	31.12	–	65.11	–
TnT	76.88	85.34	86.23	89.52	80.76	83.89	58.71	62.31	39.03	42.38	68.32	72.69
TreeTagger	72.68	81.02	83.22	87.17	73.07	81.67	51.84	58.73	29.94	40.59	62.15	69.84

It is striking how LemmaTag outclasses other taggers for morphological tagging of the Tiger corpus, reaching 93.72% accuracy for pipeline tagging, i.e. 6.73% above MarMoT in second place. Results for Latin corpora (whether ID or OD) are far closer. LemmaTag still performs reasonably well on the Capitularies, both ID and OD. But when Proiel is processed in-domain or as training set to tag Capitularies, results for LemmaTag fall to the lower end of the spectrum. An outlier worth noticing is the RDRPOSTagger, which performs best on the Proiel→Capit OD setting. As expected, the joint approach outperforms pipeline tagging in most cases, and can thus be recommended when the tagger supports it. LemmaTag can be recommended for German and it also performs well on Latin corpora, at least on the Capitularies. When accuracy for the joint approach is averaged, MarMoT obtains the best results, mainly because of its better performance on Proiel. This may be due to the fact that Proiel lacks punctuation. Future work needs to address in more detail the robustness of taggers on morphological tagging for OD settings and small corpora.

6.2 *Lemmatization*

To evaluate lemmatizers, we trained systems on the (form,lemma) pairs available in the training data. For LAT, we considered lemmatization as a tagging task, as described in Gesmundo and Samardzic (2012). We used the MarMoT tagger for this, without any additional resources. We could have used any other tagger, but MarMoT has proven to work well for POS tagging *and* can deal with large label spaces. For example, using Tiger as the training corpus for lemmatization results in 2,405 labels. This would not be suitable for Lapos or Stanford. We found (Table 21) that LemmaTag performs best in most cases (performing joint POS tagging and lemmatization); LAT is consistently better than LemmaGen, which corroborates results from Gesmundo and Samardzic (2012). We also note that ID accuracy on Tiger is higher than on TGermaCorp, but Tiger is a larger corpus than TGermaCorp (∼50,000 sentences vs. ∼9,000) and more homogeneous, while TGermaCorp contains poems, various types of literature, etc. The OD results seem pretty low, compared to ID results, as was the case for tagging. We remark that specific lemma conventions differ between the two datasets. In addition, TGermaCorp contains spelling mistakes (*Widersehen*), and historical variants (*Capital*) that are con-

Table 21: Direct lemmatization accuracy in %

	LemmaGen	MarMoT-LAT	LemmaTag
TG	91.71	91.99	**92.11**
Tiger	98.07	98.19	**98.66**
TG→Tiger	87.00	**88.57**	88.37
Tiger→TG	87.26	88.05	**90.34**
Capit	95.64	95.81	**96.13**
Proiel	**90.63**	90.29	81.85
Capit→Proiel	81.39	81.24	**82.25**
Proiel→Capit	76.28	**76.37**	49.61

ventionally lemmatized by their modern lemma forms (*Kapital*). Such cases are difficult for trained systems to handle, since they are absent from Tiger.

As Tables 22 and 23 show for the German TGermaCorp and the Latin Capitularies, the most frequent lemmatization errors by LAT and LemmaGen correlate well. Confusing the two variants of the German conjunction "dass" and "daß" (old) is partly due to the training corpus, which contains documents from different decades and centuries, and partly to genuine inconsistencies in the gold standard, which should be corrected in future editions. The impact on overall accuracy is still quite minor – normalizing these variants improves accuracy for TGermaCorp ID by about 0.1%. Table 24 illustrates how the type of lemmatization error shifts when the model is either trained based on Proiel (ID) or on the Capitularies (OD), which decreases accuracy by about 9%.

6.3 *Tagging and lemmatization*

In Section 6.1.4, we addressed the pros and cons of joint vs pipeline learning of POS and morphology tags. Here, we complete the picture by examining accuracy for lemmatization *and* tagging combined. Table 25 shows accuracy for LemmaTag (joint fine-grained tagging and lemmatization), ranging from 92.95% for lemmatization and fine-grained tagging on Tiger ID to only 30.44% on Proiel→Capit. Likewise, the loss of accuracy resulting from the extension of lemmatization by POS and morphology ranges from 5.71% on Tiger and 7.6% Capitularies to 20–25% for Proiel and the two Latin OD settings.

Table 22: Most frequent lemmatization errors with LAT, LemmaGen, and LemmaTag on TGermaCorp

LemmaGen				LAT				LemmaTag			
Form	Gold	Pred.	#	Form	Gold	Pred.	#	Form	Gold	Pred.	#
was	wer	was	71	was	wer	was	34	was	was	wer	36
einer	einer	ein	42	–	–	–	32	–	–	–	32
am	am	an	38	was	was	wer	25	rtr	–	rtr	18
eine	einer	ein	36	rtr	–	rtr	24	dessen	dessen	der	17
–	–	–	32	ap	–	ap	20	dpa	–	dpa	16
alle	alle	aller	26	dpa	–	dpa	18	was	wer	was	14
rtr	–	rtr	24	dessen	dessen	der	17	ap	–	ap	10
viele	viele	vieler	22	viele	viele	vieler	16	weiter	weit	weiter	9
Deutschen	deutsche	deutsch	22	eine	einer	ein	15	allem	aller	alle	8
ap	–	ap	21	einer	einer	ein	14	gestattet	gestatten	statten	7
uns	sich	uns	21	alle	alle	aller	13	früher	früher	früh	7
einen	einer	ein	18	einen	einer	ein	13	.	.	–	7
dpa	–	dpa	18	am	am	an	12	uns	uns	sich	7
dessen	dessen	der	17	afp	–	afp	10	uns	sich	uns	7
mich	sich	mich	17	allem	aller	alle	8	mich	sich	mich	7
eines	einer	ein	17	allen	alle	aller	8	AFP	AFP	afp	6
Den	Den	der	16	weiter	weit	weiter	8	afp	–	afp	6
lange	lang	lange	14	.	.	–	7	geeignet	eignen	geeignet	5
Polen	Pole	Polen	10	Wissen	wissen	Wissen	7	unserer	mein	unser	5
einem	einer	ein	10	früher	früher	früh	7	,	,	–	5

Table 23: Most frequent lemmatization errors for LAT, LemmaGen, and LemmaTag on Capitularies

LemmaGen				LAT				LemmaTag			
Form	Gold	Predicted	#	Form	Gold	Predicted	#	Form	Gold	Predicted	#
quod	qui	quod	157	quod	qui	quod	135	quod	qui	quod	142
quam	qui	quam	101	quam	qui	quam	82	quam	qui	quam	51
nostri	nostri	noster	58	nostri	nostri	noster	43	qua	qui	quis	42
qua	qui	quis	51	qua	qui	quis	37	nostri	nostri	noster	38
quo	qui	quo	51	quo	qui	quo	19	quam	quam	qui	24
modo	modo	modus	32	qui	quis	qui	19	bannum	bannum	bannus	21
bannum	bannum	bannus	27	fiat	fio	fiat	17	modo	modo	modus	20
primo	primo	primus	27	quod	quod	qui	17	tantum	tantus	tantum	19
tantum	tantum	tantus	22	bannum	bannum	bannus	17	qui	quis	qui	19
una	una	unus	21	quo	quo	qui	16	fiat	fio	fiat	17
qui	quis	qui	19	postea	postis	postea	16	fiant	fio	fiant	17
postea	postis	postea	19	modo	modo	modus	15	eodem	eodem	idem	14
fiant	fio	fiant	17	eodem	idem	eodem	15	nostrum	nos	noster	14
fiat	fio	fiat	17	fiant	fio	fiant	14	tantum	tantum	tantus	13
una	unus	una	16	quam	quam	qui	12	quid	quid	quis	12
bona	bonum	bonus	16	postea	postea	postis	12	factum	factum	facio	12
eodem	eodem	idem	14	tantum	tantus	tantum	12	alium	alius	prefulgidus	11
nostrum	nos	noster	14	bannum	bannus	bannum	12	maxime	magnus	magis	11
factum	factum	facio	13	maxime	magnus	magis	11	excepto	excepto	exceptus	11
quid	quid	quis	12	etc	et cetera	et	11	forte	forte	fors	10

Table 24: Most frequent lemmatization errors for LAT Proiel ID and Capitularies → Proiel OD

LAT Proiel				LAT Capit→Proiel			
Form	Gold	Predicted	#	Form	Gold	Predicted	#
quod	qui	quod	17	se	is	sui	630
quam	qui	quam	8	a	ab	a	333
minus	parum	minus	7	ac	atque	ac	291
una	una	unus	5	sibi	is	sui	210
respondit	respondeo	respondo	5	castra	castra	castrum	177
oportere	oportet	oporteo	4	sese	is	sui	153
tergum	tergum	tergus	4	quod	qui	quod	140
coeperunt	incipio	cobeo	4	uti	ut	utor	120
actis	ago	actis	4	castris	castra	castrum	99
Hi	is	hic	4	quam	qui	quam	86
Si	si	is	4	se	se	sui	69
progressus	progredior	progredo	3	minus	parum	parve	57
magis	magis	magus	3	copias	copia	copio	51
primum	primus	primum	3	Germanos	Germani	Germanus	51
iuris	ius	iur	3	Germani	Germani	Germanus	51
vocibus	vox	vocis	3	equitatu	equitatus	equitas	48
reliquit	relinquo	reliquo	3	equites	eques	equites	45
plures	multus	plures	3	Germanorum	Germani	Germanus	45
pedem	pes	pedem	3	Haeduis	Aedui	Haeduis	43
influit	influo	infelio	3	equitatum	equitatus	equitas	40

	Lemma	Lemma POS	POS morph	Lemma POS morph
TG	92.11	87.44	–	–
Tiger	98.66	97.21	93.68	92.95
TG→Tiger	88.37	82.05	–	–
Tiger→TG	90.34	83.39	–	–
Capit	96.13	94.33	90.03	88.53
Proiel	81.85	78.54	73.10	62.48
Capit→Proiel	82.25	76.22	63.43	57.40
Proiel→Capit	49.61	44.06	38.09	30.44

Table 25: Accuracy in % for combined POS tagging, morphological joint-tagging, and lemmatization, using LemmaGen

Table 26: Run-times for a partition of Tiger (ID)

	POS		Joint	
FLORS	3h16m26s	3m42s	1d7h39m56s	7m14s
Lapos	6m30s	13s		
LemmaTag (GPU)	26m7s	2s	2h23m16s	10s
LemmaTag (CPU)	18h42m57s	1m58s		
MarMoT	6m1s	10s	44m47s	38s
Mate	11m57s	21s	3h0m0s	4m4s
OpenNLP	8m10s	4s	1h17m23s	27s
RDR	20m33s	10s	21m5s	1m
Stanford	35m12s	11s		
TnT	5s	697ms	13s	2s
TreeTagger	2s	438ms	1m12s	5s

6.4 *Computing time*

Until now, we have focused on evaluating tools with regard to the accuracy achieved for tagging and lemmatization. However, when a parameter space needs to be explored to determine the optimal configuration for training a model, or when large collections of text need to be tagged, runtime cannot be neglected. Table 26 shows the time needed to train and test a partition of the Tiger corpus. Here, 45,372 tokens in 799,406 sentences were used for training, and 5,100 tokens in 88,832 sentences for testing. Please note that these numbers may vary considerably, depending on the workstation or server the processes run on, as well as the machine's current load. However, they give a first impression of what to expect. Note also that the time for testing included the time taken to load the trained files.[17] While TreeTagger takes about 2 seconds to train and less than a second to test, FLORS takes more than 3 hours to train and 3 minutes 42 seconds to test. When taking accuracy into account, MarMoT appears to be the best trade-off, as it needs only 12 minutes for training and 10 seconds for testing. Alternatively, if a GPU workstation is available, LemmaTag is a good choice.[18] Without a GPU, training and testing for LemmaTag can also be done on CPUs. But for CPU-based POS tagging, we measured a training time of over 18 hours, with a test time

[17] LemmaTag performs training and testing in one process chain, so that the model files would have already been loaded.

[18] We conducted our experiments on a workstation equipped with one NVidia GTX1080 and one GTX1060.

of about 2 minutes. So even if a GPU is used for training, testing on a CPU is still not a good option with large corpora. In order to tag large amounts of text, we recommend splitting the corpus into chunks and processing them separately – either by separate processes or by multi-threading within one application. In this regard all taggers (to our knowledge) suffer from the problem of not being thread-safe, such that multiple instances cannot share the same model file in memory. This means that each instance needs to load its own representation of the trained model. So, depending on the hardware available for tagging, a bottle-neck may be caused by short memory rather than by CPU issues.

Finally, we examine the time needed to train and test lemmatization based on Tiger. LemmaGen takes 6 seconds for training and less than 1 second for testing, while LAT based on MarMoT takes 2h29m9s for training and 1m47s for testing. Since accuracy results for LemmaGen and MarMoT-LAT differ only marginally, LemmaGen has a much better performance-runtime trade-off than MarMoT-LAT. LemmaTag takes 26m7s for training and 2s for testing – the same time as it takes for POS tagging, since it performs both tasks in one sweep.

7 DISCUSSION

In terms of accuracy, MarMoT, FLORS, Lapos, Mate, and particularly LemmaTag are the methods of choice when (especially) fine-grained morphological tagging is the goal. LemmaTag performs exceptionally well on German corpora but its results for Latin, especially on Proiel and Latin OD, show that it is not the unique solution that fits all cases. For POS tagging, MarMoT proves to be competitive. When trained with embeddings as a supplementary resource, it reaches up to 98.32% on Tiger, while LemmaTag achieved 98.58%. Moreover, the lexical resources as well as the word embeddings derived from unlabeled data that can be fed into MarMoT (and FLORS) make the systems more robust to change of domain, which is an immensely important aspect of real-world POS tagging. With respect to word embeddings, we note that the approach (FastText, GloVe, Mikolov, Komninos or Levy) and its parameters play an important role. The choice of parameters to compute embeddings can either improve or degrade accuracy, as shown in Table 16. In our settings, Mikolov and FastText outperform

GloVe. When dependency parsing information is available, Komninos yields better results than Levy. The question as to whether Komninos is better than Mikolov cannot be answered unambiguously, as the results are somewhat similar, depending on language and corpus. Our experiments on German corpora indicate that Komninos should be preferred over Mikolov, as long as enough parsed corpora are available to compute the embeddings.

Accuracy aside, it is also interesting to take a closer look at the time needed for training and testing (see Table 26).[19] Various aspects may have an impact on run-time. First, time depends linearly on the hardware being used. Second, processing times depend on the configuration of the taggers – especially with regard to training a new model. Finally, the taggers evaluated are implemented in different programming languages, and the implementations are not necessarily optimized for everyday field-use but for high accuracy. This means that even though the computational complexity of a given approach cannot be changed, a great deal of time can be saved by using efficient programming techniques, with extensive use of multi-threading. Both FLORS and the Stanford tagger can take hours or even days to train a model, depending on the size of the corpora and the parameters. In contrast, TnT and TreeTagger, while typically performing less well – sometimes badly – in terms of accuracy, take only a fraction of the training and testing time of the more recent generation. For a practitioner, computing time can be a critical issue. For example, when it comes to tagging large corpora, such as the entire Wikipedia, even a few seconds more per processed text can make a huge difference. So, depending on the task at hand, even the older approaches may still be attractive.

In this respect, TnT is particularly interesting: it is about as fast as TreeTagger, and its fine-grained tagging accuracy is often only marginally below that of MarMoT. For example, with only one exception, TnT is within ∼1% point of the (joint) fine-grained POS tagging accuracy of MarMoT (used without additional resources), while the TreeTagger is between 3% and 8% below MarMoT with respect to

[19] Please note that the computation times can only give a general impression, and include the time to load/save model files.

fine-grained tagging. At the same time, training time on Tiger (ID) for joint fine-grained tagging is 13s for TnT vs. 44m for MarMoT and above 1m for TreeTagger. The good results of LemmaTag on German corpora confirm the success of neural network approaches – at the cost of extensive computation time when no GPUs are available. This suggests that, in near future, tagging large corpora such as Wikipedia will require a solid GPU workstation or even a cluster.

Regarding the drop of performance in OD experiments, we note that OD experiments actually constitute a *lower bound* on performance, while ID experiments constitute an (ideal case) *upper bound*. The reason why ID experiments are not entirely reliable, for a practitioner, is that it is assumed that the test data will have the same distributional properties as the data on which a machine-learning system has been trained. This is implausible in almost all practical scenarios. On the other hand we also note that OD experiments often indicate errors that are, from a linguistic perspective, not errors at all but correspond, for example, to different *conventions* according to which different datasets have been annotated.

Similar to POS tagging, we can observe a trade-off between accuracy and run-time for lemmatization. In most cases, LemmaTag performs best, followed by MarMoT-LAT, and LemmaGen. Thus LemmaTag is the best solution if GPUs are available and processing time is less of an issue. When having to decide between LemmaGen and MarMoT-LAT, the latter appears to be the best choice. But for in-domain settings the difference is only marginal – the largest gap of 1.57% can be observed when TGermaCorp is used to train a model for lemmatization of Tiger. On the other hand, MarMoT-LAT takes 107 seconds for lemmatization of Tiger (ID), whereas LemmaGen takes only 5 seconds. So the choice of lemmatizer depends on the use-case. If processing time is not an issue or if there is enough hardware to scale on, MarMoT-LAT would be a good choice. Since LemmaGen's accuracy is at a similar level, even outperforming LAT in some cases, it can be considered a good option to lemmatize large corpora.

In a nutshell, we conclude that the performance of taggers can be raised by including distributional information (about paradigmatic word associations), as computed by word embeddings or by means of (still mostly handcrafted) lexicons. It is no surprise that this relates especially to OD scenarios. However, it also means that, especially in the

case of low-resource languages, for which lexicons are rarely available, neural network models are the first choice when trying to improve tagging results. In scenarios of processing historical texts, either lexical or procedural information about alternative spellings would be an alternative informational resource that may also help computing word embeddings at a more abstract level (of super-lemmas instead of lemmas alone).

Our study has also demonstrated the computational effort necessary to train and to evaluate new taggers – especially in the light of the ever-increasing size of annotated corpora that can be used for training and testing. In order to capture this effort, one may think of a *meta-learner* that takes as input (1) taggers, (2) learning scenarios (e.g., fine-grained/coarse-grained), (3) annotated corpora, and (4) additional resources (embeddings or lexica), in order to update the desired evaluation. One may think of a toolbox for building large-scale evaluation studies allowing for laborious (hyper-)parameter studies in terms of big data-experiments. Ideally, this would function *out of the box* so that newly available annotation data could be rapidly used to retrain taggers. Currently, comparative studies are still very laborious rather than being easily manageable.

8 CONCLUSION

We conducted a study of tagging for German and (classical as well as medieval) Latin texts by examining a range of older taggers in comparison with more recent ones. We experimented with coarse-grained as well as fine-grained POS tagging, and with lemmatization. Our findings highlight the improvements achieved by the most recent tagger developments. LemmaTag, in particular, performed best in most of the tasks considered here. We also show that out-of-domain (OD) tagging leads to a considerable loss in tagging accuracy. The same is true when we consider pipeline learning of inflectional categories. These findings hint at the need to further develop taggers, possibly by extending feature space (e.g., by morphological, syntactic, or even semantic features). However, our experiments also show that such an extension may lead to a considerable increase in training and operating time, and thus may be problematic in the case of time-critical scenarios. Last but not least, we evaluated three lemmatizers. Here

also, LemmaTag performs best in most cases, followed by LAT. Our experiments show once more that accuracy drops significantly if the lemmatizer is applied OD. As before, this is a good argument for further developments in this area of NLP, in particular, to address domain adaptation.

OPEN SOURCE

Models for taggers evaluated here are available online.[20] This includes all annotated corpora, as far as license terms allow.

REFERENCES

Bernd BOHNET and Joakim NIVRE (2012), A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing, in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1455–1465, Association for Computational Linguistics, Jeju Island, Korea, http://www.aclweb.org/anthology/D12-1133.

Piotr BOJANOWSKI, Edouard GRAVE, Armand JOULIN, and Tomas MIKOLOV (2017), Enriching word vectors with subword information, *Transactions of the Association for Computational Linguistics*, 5:135–146, ISSN 2307-387X.

Sabine BRANTS, Stefanie DIPPER, Peter EISENBERG, Silvia HANSEN-SCHIRRA, Esther KÖNIG, Wolfgang LEZIUS, Christian ROHRER, George SMITH, and Hans USZKOREIT (2004), TIGER: Linguistic interpretation of a German corpus, *Research on Language and Computation*, 2(4):597–620, doi:10.1007/s11168-004-7431-3.

Thorsten BRANTS (2000), TnT: A statistical part-of-speech tagger, in *Proceedings of the Sixth Conference on Applied Natural Language Processing*, ANLC '00, pp. 224–231, Association for Computational Linguistics, Stroudsburg, PA, USA, doi:10.3115/974147.974178, http://dx.doi.org/10.3115/974147.974178.

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

Paul COMPTON and Bob JANSEN (1988), Knowledge in context: A strategy for expert system maintenance, *Proceedings of the 2nd Australian Joint Artificial Intelligence Conference*, pp. 292–306.

Gregory CRANE (1991), Generating and parsing classical Greek, *Literary and Linguistic Computing*, 6(4):243–245, doi:10.1093/llc/6.4.243, http://dx.doi.org/10.1093/llc/6.4.243.

[20] https://www.texttechnologylab.org/applications/corpora/

Hannes DOHRN and Dirk RIEHLE (2011), Design and implementation of the Sweble Wikitext Parser: Unlocking the structured data of Wikipedia, in *Proceedings of the 7th International Symposium on Wikis and Open Collaboration*, WikiSym '11, pp. 72–81, ACM, New York, NY, USA, ISBN 978-1-4503-0909-7, doi:10.1145/2038558.2038571, http://doi.acm.org/10.1145/2038558.2038571.

Markus DREYER, Jason SMITH, and Jason EISNER (2008), Latent-variable modeling of string transductions with finite-state methods, in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pp. 1080–1089, Association for Computational Linguistics, Honolulu, Hawaii, http://www.aclweb.org/anthology/D08-1113.

Greg DURRETT and John DENERO (2013), Supervised learning of complete morphological paradigms, in *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9–14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pp. 1185–1195.

Steffen EGER (2015), Designing and comparing G2P-type lemmatizers for a morphology-rich language, in *Systems and Frameworks for Computational Morphology – Fourth International Workshop, SFCM 2015, Stuttgart, Germany, September 17–18, 2015, Proceedings*, pp. 27–40, doi:10.1007/978-3-319-23980-4_2.

Steffen EGER, Tim VOR DER BRÜCK, and Alexander MEHLER (2015), Lexicon-assisted tagging and lemmatization in Latin: A comparison of six taggers and two lemmatization methods, in *Proceedings of the 9th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH 2015)*, pp. 105–113, Beijing, China.

Joseph L. FLEISS (1971), Measuring nominal scale agreement among many raters, *Psychological Bulletin*, 76:378–382.

Andrea GESMUNDO and Tanja SAMARDZIC (2012), Lemmatisation as a tagging task, in *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8–14, 2012, Jeju Island, Korea – Volume 2: Short Papers*, pp. 368–372.

Birgit HAMP and Helmut FELDWEG (1997), GermaNet – a lexical-semantic net for German, in *In Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pp. 9–15.

Dag Trygve Truslew HAUG and Marius JØHNDAL (2008), Creating a parallel treebank of the old Indo-European Bible translations, in *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*.

Wahed HEMATI, Tolga USLU, and Alexander MEHLER (2016), TextImager: a Distributed UIMA-based system for NLP, in *Proceedings of the COLING 2016 System Demonstrations*, pp. 59–63, Federated Conference on Computer Science and Information Systems.

Verena HENRICH and Erhard HINRICHS (2010), GernEdiT – The GermaNet editing tool, in Nicoletta CALZOLARI, Khalid CHOUKRI, Bente MAEGAARD, Joseph MARIANI, Jan ODIJK, Stelios PIPERIDIS, Mike ROSNER, and Daniel TAPIAS, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, European Language Resources Association (ELRA), Valletta, Malta, ISBN 2-9517408-6-7.

Mans HULDEN, Markus FORSBERG, and Malin AHLBERG (2014), Semi-supervised learning of morphological paradigms and lexicons, in *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 569–578, Association for Computational Linguistics, Gothenburg, Sweden, doi:10.3115/v1/E14-1060, `https://www.aclweb.org/anthology/E14-1060`.

Matjaž JURŠIČ, Igor MOZETIČ, Tomaž ERJAVEC, and Nada LAVRAČ (2010), LemmaGen: Multilingual lemmatisation with induced ripple-down rules., *J. UCS*, 16(9):1190–1214, `http://dblp.uni-trier.de/db/journals/jucs/jucs16.html#JursicMEL10`.

Bernhard JUSSEN, Alexander MEHLER, and Alexandra ERNST (2007), A corpus management system for historical semantics, *Sprache und Datenverarbeitung. International Journal for Language Data Processing*, 31(1–2):81–89.

Alexandros KOMNINOS and Suresh MANANDHAR (2016), Dependency based embeddings for sentence classification tasks, in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1490–1500, Association for Computational Linguistics, San Diego, California, doi:10.18653/v1/N16-1175, `https://www.aclweb.org/anthology/N16-1175`.

Daniel KONDRATYUK, Tomás GAVENCIAK, Milan STRAKA, and Jan HAJIC (2018), LemmaTag: Jointly tagging and lemmatizing for morphologically-rich languages with BRNNs, *CoRR*, abs/1808.03703:4921–4928.

Klaus KRIPPENDORFF (1980), *Content analysis*, volume 5 of *The SAGE KommTexT Series*, SAGE Publications, Beverly Hills and London.

Matthieu LABEAU, Kevin LÖSER, and Alexandre ALLAUZEN (2015), Non-lexical neural architecture for fine-grained pos tagging, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 232–237, Association for Computational Linguistics, Lisbon, Portugal, `http://aclweb.org/anthology/D15-1025`.

Omer LEVY and Yoav GOLDBERG (2014), Dependency-based word embeddings, in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pp. 302–308.

Andy LÜCKING, Armin HOENEN, and Alexander MEHLER (2016), TGermaCorp – A (digital) humanities resource for (computational) linguistics, in *Proceedings of the 10th International Conference on Language Resources and Evaluation*, LREC 2016, pp. 4271–4277.

Christopher D. MANNING, Mihai SURDEANU, John BAUER, Jenny FINKEL, Steven J. BETHARD, and David MCCLOSKY (2014), The Stanford CoreNLP natural language processing toolkit, in *Association for Computational Linguistics (ACL) System Demonstrations*, pp. 55–60, http://www.aclweb.org/anthology/P/P14/P14-5010.

Barbara MCGILLIVRAY, Marco PASSAROTTI, and Paolo RUFFOLO (2009), The Index Thomisticus treebank project: Annotation, parsing and valency lexicon, *TAL*, 50(2):103–127, http://atala.org/IMG/pdf/TAL-2009-50-2-04-McGillivray.pdf.

Alexander MEHLER, Tim VOR DER BRÜCK, Rüdiger GLEIM, and Tim GEELHAAR (2015), Towards a network model of the coreness of texts: An experiment in classifying Latin texts using the TTLab Latin tagger, in Chris BIEMANN and Alexander MEHLER, editors, *Text Mining: From Ontology Learning to Automated text Processing Applications*, Theory and Applications of Natural Language Processing, pp. 87–112, Springer, Berlin/New York.

Tomas MIKOLOV, Ilya SUTSKEVER, Kai CHEN, Greg CORRADO, and Jeff DEAN (2013), Distributed representations of words and phrases and their compositionality, in *Advances in Neural Information Processing Systems 26*, pp. 3111–3119, Curran Associates, Inc., http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Stefano MINOZZI (2008), La costruzione di una base di conoscenza lessicale per la lingua latina: LatinWordnet, http://hdl.handle.net/11562/324939.

Thomas MÜLLER, Ryan COTTERELL, Alexander M. FRASER, and Hinrich SCHÜTZE (2015), Joint lemmatization and morphological tagging with Lemming, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17–21, 2015*, pp. 2268–2274.

Thomas MÜLLER, Helmut SCHMID, and Hinrich SCHÜTZE (2013), Efficient higher-order CRFs for morphological tagging, in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 322–332, Association for Computational Linguistics, Seattle, Washington, USA, http://www.aclweb.org/anthology/D13-1032.

Dat Quoc NGUYEN, Dai Quoc NGUYEN, Dang Duc PHAM, and Son Bao PHAM (2014), RDRPOSTagger: A ripple down rules-based part-of-speech tagger, in *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 17–20, Association for Computational Linguistics, Gothenburg, Sweden, http://www.aclweb.org/anthology/E14-2005.

Dat Quoc NGUYEN, Dai Quoc NGUYEN, Dang Duc PHAM, and Son Bao PHAM (2016), A robust transformation-based learning approach using ripple down

rules for part-of-speech tagging, *AI Communications*, 29(3):409–422, http://dx.doi.org/10.3233/AIC-150698.

Garrett NICOLAI, Colin CHERRY, and Grzegorz KONDRAK (2015), Inflection generation as discriminative string transduction, in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 922–931, Association for Computational Linguistics, Denver, Colorado, http://www.aclweb.org/anthology/N15-1093.

Marco PASSAROTTI (2015), What you can do with linguistically annotated data. From the Index Thomisticus to the Index Thomisticus Treebank, in *Reading Sacred Scripture with Thomas Aquinas. Hermeneutical Tools, Theological Questions and New Perspectives*, pp. 3–44, Brepols.

Marco PASSAROTTI, Marco BUDASSI, Eleonora LITTA, and Paolo RUFFOLO (2017), The Lemlat 3.0 package for morphological analysis of Latin, in *Proceedings of the NoDaLiDa 2017 workshop on processing historical language*, 133, pp. 24–31, Linköping University Electronic Press, Linköpings universitet, ISSN 1650-3740.

Jeffrey PENNINGTON, Richard SOCHER, and Christopher D. MANNING (2014), GloVe: Global vectors for word representation, in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, http://www.aclweb.org/anthology/D14-1162.

Adwait RATNAPARKHI (1996), A maximum entropy model for part-of-speech tagging, in *Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia, Pennsylvania.

Toni RIETVELD and Roeland VAN HOUT (1993), *Statistical techniques for the study of language and language behaviour*, Mouton de Gruyter, Amsterdam.

Anne SCHILLER, Simone TEUFEL, Christine STÖCKERT, and Christine THIELEN (1999), Guidelines für das Tagging deutscher Textcorpora mit STTS (kleines und großes Tagset), Technical report, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Helmut SCHMID (1994), Probabilistic part-of-speech tagging using decision trees, in *International Conference on New Methods in Language Processing*, pp. 44–49, Manchester, UK.

Tobias SCHNABEL and Hinrich SCHÜTZE (2014), FLORS: Fast and simple domain adaptation for part-of-speech tagging, *TACL*, 2:15–26, https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/183.

Carsten SCHNOBER, Steffen EGER, Erik-Lân DO DINH, and Iryna GUREVYCH (2016), Still not there? Comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks, in *Proceedings of COLING 2016, the 26th International Conference on Computational*

Linguistics: Technical Papers, pp. 1703–1714, The COLING 2016 Organizing Committee, Osaka, Japan.

Uwe SPRINGMANN, Helmut SCHMID, and Dietmar NAJOCK (2016), LatMor: A Latin finite-state morphology encoding vowel quantity, *Open Linguistics*, 2(1):386–392, doi:10.1515/opli-2016-0019.

Kristina TOUTANOVA, Dan KLEIN, Christopher D. MANNING, and Yoram SINGER (2003), Feature-rich part-of-speech tagging with a cyclic dependency network, in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology – Volume 1*, NAACL '03, pp. 173–180, Association for Computational Linguistics, Stroudsburg, PA, USA, doi:10.3115/1073445.1073478, http://dx.doi.org/10.3115/1073445.1073478.

Yoshimasa TSURUOKA, Yusuke MIYAO, and Jun'ichi KAZAMA (2011), Learning with lookahead: Can history-based models rival globally optimized models?, in *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL '11, pp. 238–246, Association for Computational Linguistics, Stroudsburg, PA, USA, ISBN 978-1-932432-92-3, http://dl.acm.org/citation.cfm?id=2018936.2018964.

Tim VOR DER BRÜCK and Alexander MEHLER (2016), TLT-CRF: A lexicon-supported morphological tagger for Latin based on conditional random fields, in Nicoletta CALZOLARI, Khalid CHOUKRI, Thierry DECLERCK, Sara GOGGI, Marko GROBELNIK, Bente MAEGAARD, Joseph MARIANI, Helene MAZO, Asuncion MORENO, Jan ODIJK, and Stelios PIPERIDIS, editors, *Proceedings of the tenth international conference on language resources and evaluation (LREC 2016)*, European Language Resources Association (ELRA), Paris, France, ISBN 978-2-9517408-9-1.

Peilu WANG, Yao QIAN, Frank K. SOONG, Lei HE, and Hai ZHAO (2015), Part-of-speech tagging with bidirectional long short-term memory recurrent neural network, *CoRR*, abs/1510.06168, http://arxiv.org/abs/1510.06168.

Wenpeng YIN, Tobias SCHNABEL, and Hinrich SCHÜTZE (2015), Online updating of word representations for part-of-speech tagging, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1329–1334, Association for Computational Linguistics, Lisbon, Portugal, http://aclweb.org/anthology/D15-1155.

This work is licensed under the Creative Commons Attribution 3.0 Unported License.
http://creativecommons.org/licenses/by/3.0/