Lemmatization and morphological tagging in German and Latin: A comparison and a survey of the state-of-the-art

Steffen Eger[1], **Rüdiger Gleim**[2], **Alexander Mehler**[2]
[1] Ubiquitous Knowledge Processing Lab, Technische Universität Darmstadt, Germany
[2]Text Technology Lab, Goethe Universität Frankfurt am Main, Germany
eger@ukp.informatik.tu-darmstadt.de,{gleim,mehler}@em.uni-frankfurt.de

Abstract

This paper relates to the challenge of morphological tagging and lemmatization in morphologically rich languages by example of German and Latin. We focus on the question what a practitioner can expect when using state-of-the-art solutions out of the box. Moreover, we contrast these with old(er) methods and implementations for POS tagging. We examine to what degree recent efforts in tagger development pay out in improved accuracies — and at what cost, in terms of training and processing time. We also conduct in-domain vs. out-domain evaluation. Out-domain evaluations are particularly insightful because the distribution of the data which is being tagged by a user will typically differ from the distribution on which the tagger has been trained. Furthermore, two lemmatization techniques are evaluated. Finally, we compare pipeline tagging vs. a tagging approach that acknowledges dependencies between inflectional categories.

Keywords: morphological tagging, lemmatization, morphologically rich languages

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, or semantic analysis. In morphologically rich languages such as German and Latin, both problems are non-trivial due to the variability of lexical forms. This results both in large tagsets for POS tagging — which list such inflectional categories as `case`, `gender`, `degree`, etc., besides 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 the two languages mentioned. Our survey includes both older, such as the TreeTagger (Schmid, 1994) and TnT (Brants, 2000), and more modern approaches to tagging and lemmatization. Although our expectation is clearly that technology steadily improves with time, it is apropri not obvious how large the gap between older and more modern approaches is, and also what the ordering of the most recent generation of systems is. We test our systems under the following requirements:[1]

- Ideally, we would want a learned system to perform well on the distribution (a specific text genre, historical language variant, etc.) on which it has been trained (*in-domain* (ID)) but also to perform decently on corpora of similar but different genres, registers, language varieties, etc. (*out-domain* (OD)).
- Since coarse-grained POS tagging may be insufficient for linguistic applications and unsatisfactory for practitioners, we expect a system to perform well on fine-grained morphological tagging, not only on coarse-grained POS labels.
- Finally, run times of systems may be of considerable interest for practitioners. Therefore, we include both training and testing time estimations of the different techniques.

[1]An ideal system can also make use of the fact that there are strong dependencies between POS tagging and lemmatization, which should substantially improve its performance relative to approaches where the tasks are treated independently (Müller et al., 2015).

2. Lemmatization

We view *lemmatization* as the problem of transforming a word form into its canoncial form, or lemma. In a machine learning context,[2] lemmatization has e.g. been considered as a character-level string transduction process (Dreyer et al., 2008; Nicolai et al., 2015; Eger, 2015), a prefix and suffix transformation problem (Jursic et al., 2010; Gesmundo and Samardzic, 2012) or as a pattern matching task (Durrett and DeNero, 2013; Ahlberg et al., 2014). 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 slower to learn and typically consider the lemmatization problem in isolation, ignoring contextual word form cues.[3]

In this work, we experiment with two approaches to lemmatization, both based on prefix and suffix transformations. **LemmaGen** (Jursic et al., 2010) learns 'ripple down rules' (Compton and Jansen, 1988), that is, tree-like decision

[2]Alternatively, lemmatization can also be implemented with the help of a lexicon. Problematic about lexicons is not only that they are hard to acquire but also that performance is usually comparatively low: lexicons cannot discriminate between alternative readings (being unaware of the distributions of forms in real text) and they cannot store an infinite number of words. However, a lexicon could typically 'assist' a learned system, e.g., via features that trigger on whether a form occurs in the lexicon (e.g., in a similar manner as outlined below).

[3]In addition, we found in preliminary experiments that for real-world lemmatization, where forms in texts must be lemmatized whose distribution is marked by many irregular forms, simpler systems, such as prefix and suffix transformation systems, may be competitive with the more sophisticated string transducers.

structures, from pairs of strings. Rule conditions are suffixes of word forms, and rule consequents are transformations which 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, the transformation of German verb form *gespielt* into its lemma *spielen* is encoded by the 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 allows to view lemmatization as a classification problem where the size of the output space is relatively small, on the orders of at most hundreds or thousands of labels. Moreover, lemmatization can then also be treated as a sequence labeling problem, where dependence between subsequent labels may be taken into account.

3. POS tagging

POS tagging (or sequence labeling) has witnessed several milestones such as 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), or the concept of the *margin* introduced in SVMs. The most recent class of taggers is characterized by the availability to include *word representations* learned from *unlabeled* data, the possibility to apply feature-rich models to problems with large output spaces, and/or by making use of *deep* (rather than *shallow*) models such as neural networks that can in addition function without handcrafting features.

In this work, we consider the following part-of-speech tagging systems, listed by the order of their year of publication: **TreeTagger** (Schmid, 1994), **TnT** (Brants, 2000), **Stanford tagger** (Toutanova et al., 2003), **Lapos** (Tsuruoka et al., 2011), **Mate** (Bohnet and Nivre, 2012). We also include the **OpenNLPTagger**, an official Apache project.[4] For these systems, we refer to the original works for descriptions. Among the most recent generation of taggers, we consider the **MarMoT** (Müller et al., 2013) tagger, 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. **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.[5] In principle, the vector representations of words are the same for known and unknown words, whence 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.[6]

In Table 1, we list some of the properties of our surveyed taggers. 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.

4. Datasets

corpus	language	sentences	tokens
Tiger	German	50,472	888,238
TGermaCorp	German	7,274	123,742
Capitularies	Latin	15,572	481,578
Proiel	Latin	1,147	22,280

Table 2: Statistics of corpa used in the experiments.

For German, we train and test on the Tiger corpus (Brants et al., 2004) and TGermaCorp (Lücking et al., 2016). For Latin, we similarly use the capitularies (Mehler et al., 2015; Eger et al., 2015) and the Proiel corpus (Haug and Jøhndal, 2008).[7] See Table 2 for details. In general, for ID experiments, we perform 3-fold random subset validation with a 90%/10% split on one of the corpora for each language. For the OD experiments, we use the entire corpora, per language, as training and test sets, respectively. As additional resources, we include the following:

- We train CBOW **word embeddings** using word2vec (Mikolov et al., 2013) on the German Wikipedia and, for Latin, on the Patrologia Latina[8].
- As **lexicons**, for German, we extract 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[10] from 2015-09-01. This is, according to our experience, more accurate than trying to parse the MediaWiki sources directly and it saves bandwith on the official Wiktionary servers. For the current experiments

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

[5] The implementation of FLORS includes language (= English) specific features. This is expected to decrease its performance on the Latin and German datasets we consider.

[6] 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 as the FLORS tagger. However, we could not make this tagger perform on-par with the other taggers surveyed. One reason for this was its immense runtime — on the orders of several days on a single training fold — so that we could not sufficiently experiment with its parameters.

[7] We used a random subset of Proiel for which tag labels had been manually synchronized with those of the capitularies.

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

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

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

	User-defined features	Large output spaces	external resources	label dependencies
FLORS		✓	✓	
Lapos				✓
MarMoT	✓	✓	✓	✓
Mate		✓		✓
OpenNLP				✓
Stanford	✓			✓
TnT		✓		✓
TreeTagger		✓		

Table 1: Systems and selected properties.

detailed below, we used a lexicon containing 67,034 lemmas and 1,817,735 syntactic words. This lexicon consists of nouns, proper nouns, verbs and adjectives. For Latin, we make use of Collex-LA (Mehler et al., 2015).

Among our taggers, only the MarMoT tagger can make use of these additional resources. Finally, we feed the first 500,000 sentences from German Wikipedia to the FLORS tagger as additional resource from which to induce word representations.[11]

5. Experiments

In this section, we detail our experiments. For POS tagging, we pursue two sets of experiments. First, we train and test on each subcategory (`pos`, `case`, `gender`, etc.) independently (*pipeline learning*). Subsequently, we train on complex tag labels, which encode the different subcategories ('*joint learning*'). Of course, joint learning is more accurate in principle — though not always computationally feasible, depending on the tagger — since it better captures the dependencies between the different categories (a noun does not admit a comparison degree, for example). We note that we generally perform no *hyperparameter optimization*, which may be considered an art in itself. Rather, we treat the taggers as block boxes and use default parametrizations.[12]

5.1. POS Tagging

5.1.1. Pipeline learning

We start by focussing on the results for `pos` (see Table 3) and `case` (Table 6), as tagging `pos` is the classical instance of (coarse-grained) POS tagging, and `case` is the most difficult category for both German and Latin. Here we run all taggers *without additional resources*.

For `pos`, MarMoT and FLORS perform best and for `case`, MarMoT and Lapos achieve best accuracies, although the performances of FLORS and Mate are not substantially worse. It is interesting to observe that the accuracies of the feature-poor models TnT and TreeTagger are substantially worse on `case` (up to 11 percentage points), relative to the best taggers, than they are on `pos` (less than 2 percentage points). Concerning run times, TnT and TreeTagger are extremely fast, while feature-richer models take substantially longer to both train and test. This hints at an interesting time-accuracy tradeoff.

The results also demonstrate the significant loss of accuracy when the training and test datasets stem from different domains (OD): the accuracies typically drop about 8-10%, across all taggers, and even beyond 30% points in particular cases. Table 4 shows, however, how the results for MarMoT improve for the OD tasks when we supply the system with additional resources in the form of embeddings and lexicon predictions. Improvements are particularly striking when training data is scarce, as is the case for the Proiel dataset. For example, accuracy in the scenario Proiel→Capit improves from 63.13% to 71.17%, when both embeddings and the lexicon is included.

	+Embeddings	+Embeddings+Lexicon
Capit→Proiel	88.40	88.79
Proiel→Capit	66.28	71.17

Table 4: Accuracy for `pos` tagging using MarMoT on Latin OD tasks.

Table 7 depicts the accuracies of all systems when fine-grained morphological tagging is the goal (under the current pipeline approach). Here, all subcategories have to match with the gold-standard. The top three systems in this setting are MarMoT, Lapos, and FLORS, while the worst two systems are TnT and the TreeTagger. Overall, differ-

	training	testing
FLORS	5:11:55	3:55
Lapos	6:44	0:13
MarMoT (Em.&L.)	5:00 (16:42)	0:10 (1:12)
Mate	30:12	0:36
OpenNLP	11:02	0:06
Stanford	27:08:17	0:08
TnT	0:02	0:01
TreeTagger	0:01	0:01

Table 5: elapsed time (in hh:mm:ss) for `pos` tagging and testing of a Tiger subset. Training and test data consists of 45,424 and 5,048 sentences, respectively.

[11] Due to memory limitations, we could not include the full Wikipedia.

[12] We note that hyperparameter optimization would generally have hardly been feasible in our case, given the running times of the systems we consider and the amount of experiments we conduct.

	FLORS	Lapos	MarMoT	Mate	OpenNLP	Stanford	TnT	TreeTagger
TG	92.36	93.02	**93.63**	93.06	91.75	91.05	92.39	92.40
Tiger	97.75	97.89	**98.04**	97.93	96.85	97.31	97.32	97.25
TG→ Tiger	89.09	89.62	**89.86**	88.22	86.65	84.02	88.28	87.74
Tiger→ TG	**90.59**	90.18	89.83	90.13	88.66	88.37	89.49	89.29
Capit	95.33	**95.97**	95.93	95.71	94.73	94.86	95.31	95.04
Proiel	92.80	95.23	**95.42**	94.93	93.12	90.97	93.63	93.21
Capit→ Proiel	84.50	87.34	**87.36**	86.54	82.46	81.41	85.62	85.34
Proiel→ Capit	62.39	63.60	63.13	**64.05**	62.38	54.43	62.92	61.23

Table 3: `pos` accuracy in %

	Tiger	Capit
FLORS	92.36	94.28
Lapos	**92.58**	94.71
MarMoT	92.46	**94.76**
Mate	92.29	94.62
OpenNLP	91.92	93.53
Stanford	90.14	93.18
TnT	84.38	92.97
TreeTagger	81.74	88.46

Table 6: `case` accuracy in %

	Tiger	Capit
FLORS	86.26	86.94
Lapos	85.86	**88.63**
MarMoT	**86.79**	88.47
Mate	83.38	87.73
OpenNLP	81.77	84.11
Stanford	81.94	85.53
TnT	76.87	86.00
TreeTagger	72.68	83.11

Table 7: all categories (pipeline) - accuracy in %

ences across tagging tasks accumulate to an up to 14% difference on the fine-grained morphological tagging task between systems.

5.1.2. Joint learning

In this subsection, we train systems on complex POS label, which comprise the different subcategories. All systems gain now significantly (Table 8). The ordering of systems in terms of accuracy is virtually the same as for pipeline learning, however. In Table 9, we zoom in on the MarMoT tagger as we provide additional resources to the system: adding word embeddings increases accuracy on morphological tagging from 90.56% to 90.91%; including the Wiktionary lexicon further increases this number to 91.04%. Table 10 shows analogous improvements for the FLORS tagger as we supply the tagger with the unlabeled Wiktionary data.

Table 11 summarizes the best possible performances of all systems for the fine-grained morphological tagging task. Here, MarMoT is clearly the best system and the only system surpassing the 90% accuracy threshold, for both the

	Tiger	Capit
FLORS	88.46	89.33
Lapos	x	x
MarMoT	**90.56**	**90.66**
Mate	84.91	89.73
OpenNLP	85.50	87.33
Stanford	x	x
TnT	85.34	89.41
TreeTagger	80.91	86.97

Table 8: all categories (joint) - accuracy in %

German and Latin ID datasets.

	Baseline	+Emb	+Emb+Lex
`pos`	98.01	98.16	98.20
`case`	94.45	94.52	94.53
`degree`	99.60	99.65	99.65
`gender`	96.83	97.13	97.27
`mood`	99.39	99.41	99.42
`number`	97.90	98.06	98.07
`person`	99.42	99.44	99.45
`tense`	99.45	99.48	99.49
ALL	90.56	90.91	91.04

Table 9: MarMoT-'Joint learning' — Tiger, ID

	Baseline	+Wiki
`pos`	97.35	97.50
`case`	93.30	93.46
`degree`	99.39	99.44
`gender`	95.73	95.99
`mood`	99.26	99.28
`number`	97.33	97.52
`person`	99.29	99.31
`tense`	99.31	99.34
ALL	88.46	88.81

Table 10: FLORS-'Joint learning' — Tiger, ID

	Tiger	Capit
FLORS	88.81	89.33
Lapos	85.86	88.63
MarMoT	**91.04**	**91.57**
Mate	84.91	89.73
OpenNLP	85.50	87.33
Stanford	81.94	85.53
TnT	85.34	89.41
TreeTagger	80.91	86.97

Table 11: Summarizing the best performances: accuracy in %

	TG	Tiger	TG→Tiger	Tiger→TG	Capit
LAT	91.75	98.24	87.48	87.70	95.30
LGen	91.37	98.10	85.95	87.61	95.19

Table 12: Direct lemmatization accuracy in %

5.2. Lemmatization

Here, we train systems on (form,lemma) pairs as available in the respective training data.[13] For LAT, we consider lemmatization as tagging task as in Gesmundo and Samardzic (2012). We use the MarMoT tagger for this, without any additional resources. In contrast, LemmaGen cannot learn to lemmatize words in context. We find (Table 12) that 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. This is due to the fact that Tiger is a larger database than TGermaCorp (around 50,000 sentences vs. around 7,500) and that Tiger is more homogeneous, while TGermaCorp contains poems, various sorts of literature, etc. Finally, the OD results seem pretty low, compared to ID results, similarly as for tagging. We remark here that specific lemma conventions differ across the two datasets. In addition, TGermaCorp contains spelling mistakes (*Widersehen*) and historical variants (*Capital*) that are conventionally lemmatized by their modern lemma forms (*Kapital*). Such cases are difficult for the trained systems to handle since they are absent in Tiger.

6. Discussion

In terms of accuracy, FLORS, Lapos, Mate, and particularly MarMoT are the methods of choice when (especially) fine-grained morphological tagging is the goal. Moreover, the lexical resources as well as the word embeddings derived from unlabeled data that can be fed in to MarMoT (and FLORS) make the systems more robust to change of domain, which is an immensely important aspect of real-world POS tagging. But it is also interesting to have a closer look at the elapsed time for training and testing (see Table 5).[14] TnT and TreeTagger, while typically performing worse — sometimes badly — in terms of accuracy, take only a fraction of training and testing time of the more recent tagger generation. For a practicioner, computing time can be a critical issue, however. 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.

7. Conclusion

We conducted a study of tagging German and (classical as well as medieval) Latin texts by examining a range of older taggers in comparison to younger ones. We experimented with coarse-grained as well as fine-grained POS tagging and with lemmatization. Our findings are in support of utilizing most recent achievements in the development of taggers. Especially, MarMoT performed best in most of the tasks considered here. We also show that out-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 of further developing taggers possibly by extending the 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 of training and operating time and, thus, may be problematic in the case of time-critical scenarios. Last but not least, we evaluated two lemmatizers. Here, LAT performs best, that is, a model which regards lemmatization as a contextual classification task. Our experiments show once more that accuracy drops considerably below 90% if the lemmatizer is applied OD. As before, this is a good argument for further developments in this area of NLP, in particular, for addressing domain adaptation.

8. Acknowledgements

This work has been funded by the Bundesministerium für Bildung und Forschung (BMBF) in the context of the *CompHistSem* project. We thank Barbara Job and Bianca Henrichfreise from Bielefeld University for annotating a subset of sentences of the Proiel corpus. We also thank Bernhard Jussen and Tim Geelhaar from Frankfurt University for providing the capitularies corpus.

Ahlberg, M., Forsberg, M., and Hulden, M. (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, Gothenburg, Sweden 2630 April 2014*, pages 569–578.

[13] It is also possible to train a lemmatizer for each POS class and apply the respective lemmatizer after a specific tagger has made its predictions (Eger et al., 2015). In auxiliary experiments not included here, we found this pipeline approach to be overall slightly worse than not relying on tagger outputs, most likely due to error propagation.

[14] Please note that the computation times shown in Table 5 can only give a general impression. Firstly, elapsed times depend linearly on the hardware being used. Secondly, processing times depend on the configuration of the taggers — especially with regard to training a new model. Finally, the evaluated taggers are implemented in different programming languages. This particularly adversely affects the FLORS tagger, which, evidently suffers from an inefficient implementation, see http://cistern.cis.lmu.de/flors/. However, the results indicate what a black box user may expect when running each of the implementations.

Bohnet, B. and Nivre, J. (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*, pages 1455–1465, Jeju Island, Korea, July. Association for Computational Linguistics.

Brants, S., Dipper, S., Eisenberg, P., Hansen-Schirra, S., Knig, E., Lezius, W., Rohrer, C., Smith, G., and Uszkoreit, H. (2004). Tiger: Linguistic interpretation of a german corpus. *Research on Language and Computation*, 2(4):597–620.

Brants, T. (2000). Tnt: A statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, ANLC '00, pages 224–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

Compton, P. and Jansen, R. (1988). Knowledge in context: A strategy for expert system maintenance. *Proceedings of the 2nd Australian Joint Artificial Intelligence Conference*, pages 292–306.

Dreyer, M., Smith, J., and Eisner, J. (2008). Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1080–1089, Honolulu, Hawaii, October. Association for Computational Linguistics.

Durrett, G. and DeNero, J. (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*, pages 1185–1195.

Eger, S., vor der Brück, T., and Mehler, A. (2015). Lexicon-assisted tagging and lemmatization in Latin: A comparison of six taggers and two lemmatization models. In *Proceedings of the 9th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH 2015)*, Beijing, China.

Eger, S. (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*, pages 27–40.

Gesmundo, A. and Samardzic, T. (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*, pages 368–372.

Haug, D. T. T. and Jøhndal, M. (2008). Creating a parallel treebank of the old Indo-European Bible translations. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*.

Jursic, M., Mozetic, I., Erjavec, T., and Lavrac, N. (2010). Lemmagen: Multilingual lemmatisation with induced ripple-down rules. *J. UCS*, 16(9):1190–1214.

Labeau, M., Löser, K., and Allauzen, A. (2015). Non-lexical neural architecture for fine-grained pos tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 232–237, Lisbon, Portugal, September. Association for Computational Linguistics.

Lücking, A., Hoenen, A., and Mehler, A. (2016). Tgermacorp a (digital) humanities resource for (computational) linguistics. In *Proceedings of the Tenth International Language Resources and Evaluation (LREC'16)*.

Mehler, A., vor der Brück, T., Gleim, R., and Geelhaar, T. (2015). Towards a network model of the coreness of texts: An experiment in classifying latin texts sing the TTLab Latin tagger. In Chris Biemann et al., editors, *Text Mining: From Ontology Learning to Automated text Processing Applications*, Theory and Applications of Natural Language Processing, pages 87–112. Springer, Berlin/New York.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, et al., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Müller, T., Schmid, H., and Schütze, H. (2013). Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA, October. Association for Computational Linguistics.

Müller, T., Cotterell, R., Fraser, A. M., and Schütze, H. (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*, pages 2268–2274.

Nicolai, G., Cherry, C., and Kondrak, G. (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*, pages 922–931, Denver, Colorado, May–June. Association for Computational Linguistics.

Ratnaparkhi, A. (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.

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

Schnabel, T. and Schütze, H. (2014). FLORS: fast and simple domain adaptation for part-of-speech tagging. *TACL*, 2:15–26.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (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, pages 173–180, Stroudsburg, PA, USA. Association for Computational

Linguistics.

Tsuruoka, Y., Miyao, Y., and Kazama, J. (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, pages 238–246, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yin, W., Schnabel, T., and Schütze, H. (2015). Online updating of word representations for part-of-speech tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1329–1334, Lisbon, Portugal, September. Association for Computational Linguistics.