A Simple Joint Model for Improved Contextual Neural Lemmatization

Chaitanya Malaviya[*,β] and **Shijie Wu**[*,ə] and **Ryan Cotterell**[ə,fi]
[β]Allen Institute for Artificial Intelligence
[ə]Department of Computer Science, Johns Hopkins University
[fi]Department of Computer Science and Technology, University of Cambridge
chaitanyam@allenai.org, shijie.wu@jhu.edu, rdc42@cam.ac.uk

Abstract

English verbs have multiple forms. For instance, *talk* may also appear as *talks*, *talked* or *talking*, depending on the context. The NLP task of lemmatization seeks to map these diverse forms back to a canonical one, known as the lemma. We present a simple joint neural model for lemmatization and morphological tagging that achieves state-of-the-art results on 20 languages from the Universal Dependencies corpora. Our paper describes the model in addition to training and decoding procedures. Error analysis indicates that joint morphological tagging and lemmatization is especially helpful in low-resource lemmatization and languages that display a larger degree of morphological complexity. Code and pre-trained models are available at https://sigmorphon.github.io/sharedtasks/2019/task2/.

1 Introduction

Lemmatization is a core NLP task that involves a string-to-string transduction from an inflected word form to its citation form, known as the lemma. More concretely, consider the English sentence: *The bulls are running in Pamplona*. A lemmatizer will seek to map each word to a form you may find in a dictionary—for instance, mapping *running* to *run*. This linguistic normalization is important in several downstream NLP applications, especially for highly inflected languages. Lemmatization has previously been shown to improve recall for information retrieval (Kanis and Skorkovská, 2010; Monz and De Rijke, 2001), to aid machine translation (Fraser et al., 2012; Chahuneau et al., 2013) and is a core part of modern parsing systems (Björkelund et al., 2010; Zeman et al., 2018).

However, the task is quite nuanced as the proper choice of the lemma is context dependent. For instance, in the sentence *A running of the bulls took place in Pamplona*, the word *running* is its own lemma, since, here, *running* is a noun rather than an inflected verb. Several counter-examples exist to this trend, as discussed in depth in Haspelmath and Sims (2013). Thus, a good lemmatizer must make use of some representation of each word's sentential context. The research question in this work is, then, how do we design a lemmatization model that best extracts the morpho-syntax from the sentential context?

Recent work (Bergmanis and Goldwater, 2018) has presented a system that directly summarizes the sentential context using a recurrent neural network to decide how to lemmatize. As Bergmanis and Goldwater (2018)'s system currently achieves state-of-the-art results, it *must* implicitly learn a contextual representation that encodes the necessary morpho-syntax, as such knowledge is requisite for the task. We contend, however, that rather than expecting the network to *implicitly* learn some no-

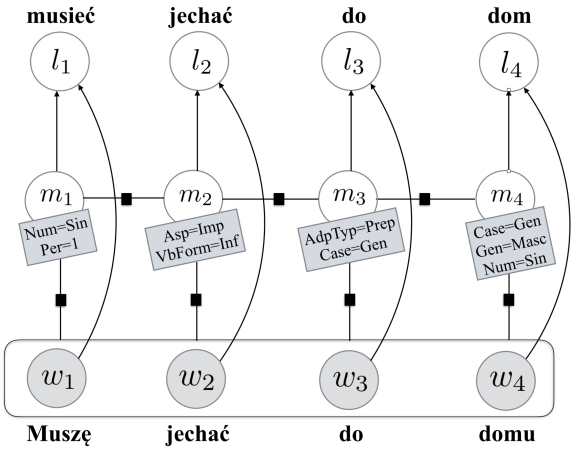

Figure 1: Our structured neural model shown as a hybrid (directed-undirected) graphical model (Koller and Friedman, 2009). Notionally, the w_i denote inflected word forms, the m_i denote morphological tags and the ℓ_i denote lemmata.

* Equal contribution. Listing order is random.

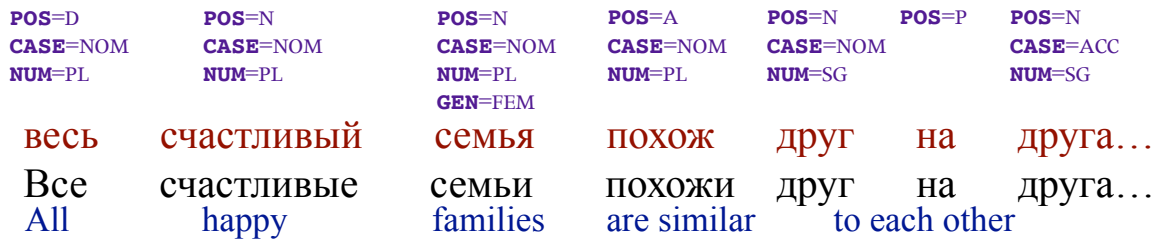

Figure 2: Example of a morphologically tagged (in purple) and lemmatized (in red) sentence in Russian using the annotation scheme provided in the UD dataset. The translation is given below (in blue).

tion of morpho-syntax, it is better to *explicitly* train a joint model to morphologically disambiguate and lemmatize. Indeed, to this end, we introduce a joint model for the introduction of morphology into a neural lemmatizer. A key feature of our model is its *simplicity*: Our contribution is to show how to stitch existing models together into a joint model, explaining how to train and decode the model. However, despite the model's simplicity, it still achieves a significant improvement over the state of the art on our target task: lemmatization.

Experimentally, our contributions are threefold. First, we show that our joint model achieves state-of-the-art results, outperforming (on average) all competing approaches on a 20-language subset of the Universal Dependencies (UD) corpora (Nivre et al., 2017). Second, by providing the joint model with gold morphological tags, we demonstrate that we are far from achieving the upper bound on performance—improvements on morphological tagging could lead to substantially better lemmatization. Finally, we provide a detailed error analysis indicating *when* and *why* morphological analysis helps lemmatization. We offer two tangible recommendations: one is better off using a joint model (i) for languages with fewer training data available and (ii) languages that have richer morphology.

Our system and pre-trained models on all languages in the latest version of the UD corpora[1][2] are released at `https://sigmorphon.github.io/sharedtasks/2019/task2/`.

2 Background: Lemmatization

Most languages (Dryer and Haspelmath, 2013) in the world exhibit a linguistic phenomenon known as **inflectional morphology**, which causes word forms to mutate according to the syntactic category of the word. The syntactic context in which the word form occurs determines which form is properly used. One privileged form in the set of inflections is called the **lemma**. We regard the lemma as a lexicographic convention, often used to better organize dictionaries. Thus, the choice of which inflected form is the lemma is motivated by tradition and convenience, e.g., the lemma is the infinitive for verbs in some Indo-European languages, rather than by linguistic or cognitive concerns. Note that the **stem** differs from the lemma in that the stem may not be an actual inflection.[3] In the NLP literature, the syntactic category that each inflected form encodes is called the **morphological tag**. The morphological tag generalizes traditional part-of-speech tags, enriching them with further linguistic knowledge such as tense, mood, and grammatical case. We call the individual key–attribute pairs **morphological attributes**.

An example of a sentence annotated with morphological tags and lemmata in context is given in Figure 2. The task of mapping a sentence to a sequence of morphological tags is known as **morphological tagging**.

Notation. Let $\mathbf{w} = w_1, \ldots, w_n$ be a sequence of n words. Each individual word is denoted as w_i. Likewise, let $\mathbf{m} = m_1, \ldots, m_n$ and $\boldsymbol{\ell} = \ell_1, \ldots, \ell_n$ be sequences of morphological tags and lemmata, respectively. We will denote the set of all tags seen in a treebank as $\mathcal{Y}$. We remark that m_i is w_i's morphological tag (e.g. [POS=N, CASE=NOM, NUM=SG] as a single label) and ℓ_i is w_i's lemma. We will denote a language's discrete alphabet of characters as Σ. Thus, we have $w_i, \ell_i \in \Sigma^*$. Furthermore, we $\mathbf{c} = c_1, \ldots, c_n$ be a vector of characters where $c_i \in \Sigma$.

[1] We compare to previously published numbers on non-recent versions of UD, but the models we release are trained on the current version (2.3).

[2] Instead of UD schema for morphological attributs, we use the UniMorph schema (Sylak-Glassman, 2016) instead. Note the mapping from UD schema to UniMorph schema is not one-to-one mapping (McCarthy et al., 2018).

[3] The stem is also often ill-defined. What is, for instance, the stem of the word *running*, is it *run* or *runn*?

3 A Joint Neural Model

The primary contribution of this paper is a joint model of morphological tagging and lemmatization. The intuition behind the joint model is simple: high-accuracy lemmatization requires a representation of the sentential context, in which the word occurs (this behind has been evinced in §1)—a morphological tag provides the precise summary of the context required to choose the correct lemma. Armed with this, we define our joint model of lemmatization and morphological tagging as:

$$p(\boldsymbol{\ell}, \mathbf{m} \mid \mathbf{w}) = p(\boldsymbol{\ell} \mid \mathbf{m}, \mathbf{w})\, p(\mathbf{m} \mid \mathbf{w}) \quad (1)$$
$$= \left(\prod_{i=1}^{n} \underbrace{p(\ell_i \mid m_i, w_i)}_{\textit{Neural Transducer}} \right) \underbrace{p(\mathbf{m} \mid \mathbf{w})}_{\textit{Neural Tagger}}$$

Figure 1 illustrates the structure of our model in the form of a graphical model. We will discuss the lemmatization factor and the morphological tagging factor following two subsections, separately. We caution the reader that the discussion of these models will be brief: Neither of these particular components is novel with respect to the literature, so the formal details of the two models is best found in the original papers. The point of our paper is to describe a simple manner to combine these existing parts into a state-of-the-art lemmatizer.

3.1 Morphological Tagger: $p(\mathbf{m} \mid \mathbf{w})$

We employ a simple LSTM-based tagger to recover the morphology of a sentence (Heigold et al., 2017; Cotterell and Heigold, 2017). We also experimented with the neural conditional random field of Malaviya et al. (2018), but Heigold et al. (2017) gave slightly better tagging scores on average and is faster to train. Given a sequence of n words $\mathbf{w} = w_1, \ldots, w_n$, we would like to obtain the morphological tags $\mathbf{m} = m_1, \ldots, m_n$ for each word, where $m_i \in \mathcal{Y}$. The model first obtains a word representation for each token using a character-level biLSTM (Graves et al., 2013) embedder, which is then input to a word-level biLSTM tagger that predicts tags for each word. Given a function cLSTM that returns the last hidden state of a character-based LSTM, first we obtain a word representation $\mathbf{u}_i$ for word w_i as,

$$\mathbf{u}_i = [\text{cLSTM}(c_1 \ldots c_n); \text{cLSTM}(c_n \ldots c_1)] \quad (2)$$

where $c_1, \ldots, c_n$ is the character sequence of the word. This representation $\mathbf{u}_i$ is then input to a word-level biLSTM tagger. The word-level biLSTM tagger predicts a tag from $\mathcal{Y}$. A full description of the model is found in Heigold et al. (2017). We use standard cross-entropy loss for training this model and decode greedily while predicting the tags during test-time. Note that greedy decoding is optimal in this tagger as there is no interdependence between the tags m_i.

3.2 A Lemmatizer: $p(\ell_i \mid m_i, w_i)$

Neural sequence-to-sequence models (Sutskever et al., 2014; Bahdanau et al., 2015) have yielded state-of-the-art performance on the task of generating morphological variants—including the lemma—as evinced in several recent shared tasks on the subject (Cotterell et al., 2016, 2017, 2018). Our lemmatization factor in eq. (1) is based on such models. Specifically, we make use of a hard-attention mechanism (Xu et al., 2015; Rastogi et al., 2016), rather than the original soft-attention mechanism. Our choice of hard attention is motivated by the performance of Makarov and Clematide (2018)'s system at the CoNLL-SIGMORPHON task. We use a nearly identical model, but opt for an exact dynamic-programming-based inference scheme (Wu et al., 2018).[4]

We briefly describe the model here. Given an inflected word w and a tag m, we would like to obtain the lemma $\ell \in \Sigma^*$, dropping the subscript for simplicity. Moreover, for the remainder of *this* section the subscripts will index into the character string ℓ, that is $\ell = \ell_1, \ldots, \ell_{|\ell|}$, where each $\ell_i \in \Sigma$. A character-level biLSTM encoder embeds w to $\mathbf{h}^{(enc)}$. The decoder LSTM produces $\mathbf{h}_j^{(dec)}$, reading the concatenation of the embedding of the previous character $\ell_{j-1} \in \Sigma$ and the tag embedding $\mathbf{h}^{(tag)}$, which is produced by an order-invariant linear function. In contrast to soft attention, hard attention models the alignment distribution explicitly.

We denote $A = \{a_1, \ldots, a_{|w|}\}^{|\ell|}$ as the set of all monotonic alignments from w to ℓ where an alignment aligns each target character ℓ_j to exactly one source character in w and for $\mathbf{a} \in A$, $a_j = i$ refers to the event that the j^{th} character of ℓ is aligned to the i^{th} character of w. We factor the probabilistic lemmatizer as,

$$p(\ell \mid m, w) = \sum_{\mathbf{a} \in A} p(\ell, \mathbf{a} \mid m, w) \quad (3)$$

[4] Our formulation differs from the work of Wu et al. (2018) in that we enforce monotonic hard alignments, rather than allow for non-monotonic alignments.

$$= \sum_{\mathbf{a} \in A} \prod_{j=1}^{|\ell|} p(\ell_j \mid a_j, \ell_{<j}, m, w) \quad (4)$$
$$p(a_j \mid a_{j-1}, \ell_{<j}, m, w)$$

$$= \sum_{\mathbf{a} \in A} \prod_{j=1}^{|\ell|} p(\ell_j \mid \mathbf{h}^{(enc)}_{a_j}, \mathbf{h}^{(dec)}_j) \quad (5)$$
$$p(a_j \mid a_{j-1}, \mathbf{h}^{(enc)}, \mathbf{h}^{(dec)}_j)$$

The summation is computed with dynamic programming—specifically, using the forward algorithm for hidden Markov models (Rabiner, 1989). $p(\ell_j \mid \mathbf{h}^{(enc)}_{a_j}, \mathbf{h}^{(dec)}_j)$ is a two-layer feed-forward network followed by a softmax. The transition $p(a_j \mid a_{j-1}, \mathbf{h}^{(enc)}, \mathbf{h}^{(dec)}_j)$ is the multiplicative attention function with $\mathbf{h}^{(enc)}$ and $\mathbf{h}^{(dec)}_j$ as input. To enforce monotonicity, $p(a_j \mid a_{j-1}) = 0$ if $a_j < a_{j-1}$.

3.3 Decoding

We consider two manners, by which we decode our model. The first is a greedy decoding scheme. The second is a **crunching** (May and Knight, 2006) scheme. We describe each in turn.

Greedy Decoding. In the greedy scheme, we select the best morphological tag sequence

$$\mathbf{m}^\star = \text{argmax}_{\mathbf{m}} \log p(\mathbf{m} \mid \mathbf{w}) \quad (6)$$

and then decode each lemmata

$$\ell^\star_i = \text{argmax}_\ell \log p(\ell \mid m^\star_i, w_i) \quad (7)$$

Note that we slightly abuse notation since the argmax here is *approximate*: exact decoding of our neural lemmatizer is hard. This sort of scheme is also referred to as pipeline decoding.

Crunching. In the crunching scheme, we first extract a k-best list of taggings from the morphological tagger. For an input sentence $\mathbf{w}$, call the k-best tags for the i^{th} word $\mathcal{K}(w_i)$. Crunching then says we should decode in the following manner

$$\ell^\star_i = \text{argmax}_\ell \quad (8)$$
$$\log \sum_{m_i \in \mathcal{K}(w_i)} p(\ell \mid m_i, w_i)\, p(m_i \mid \mathbf{w})$$

Crunching is a tractable heuristic that approximates true joint decoding[5] and, as such, we expect it to outperform the more naïve greedy approach.

[5]True joint decoding would sum over all possible morphological tags, rather than just the k-best. While this is tractable in our setting in the sense that there are, at, most, 1662 morphological tags (in the case of Basque), it is significantly slower than using a smaller k. Indeed, the probability distribution that morphological taggers learn tend to be peaked to the point that considering improbable tags is not necessary.

3.4 Training with Jackknifing

In our model, a simple application of maximum-likelihood estimation (MLE) is unlikely to work well. The reason is that our model is a discriminative directed graphical model (as seen in Figure 1) and, thus, suffers from **exposure bias** (Ranzato et al., 2015). The intuition behind the poor performance of MLE is simple: the output of the lemmatizer depends on the output of the morphological tagger; as the lemmatizer has only ever seen correct morphological tags, it has never learned to adjust for the errors that will be made at the time of decoding. To compensate for this, we employ **jackknifing** (Agić and Schluter, 2017), which is standard practice in many NLP pipelines, such as dependency parsing.

Jackknifing for training NLP pipelines is quite similar to the oft-employed cross-validation. We divide our training data into κ splits. Then, for each split $i \in \{1, \ldots, \kappa\}$, we train the morphological tagger on the i^{th} split, and then decode it, using either greedy decoding or crunching, on the remaining $(\kappa - 1)$ splits. This technique helps avoid exposure bias and improves the lemmatization performance, which we will demonstrate empirically in §4. Indeed, the model is quite ineffective without this training regime. Note that we employ jackknifing for both the greedy decoding scheme and the crunching decoding scheme.

4 Experimental Setup

4.1 Dataset

To enable a fair comparison with Bergmanis and Goldwater (2018), we use the Universal Dependencies Treebanks (Nivre et al., 2017) for all our experiments. Following previous work, we use v2.0 of the treebanks for all languages, except Dutch, for which v2.1 was used due to inconsistencies in v2.0. The standard splits are used for all treebanks.

4.2 Training Setup and Hyperparameters

For the morphological tagger, we use the baseline implementation from Malaviya et al. (2018). This implementation uses an input layer and linear layer dimension of 128 and a 2-layer LSTM with a hidden layer dimension of 256. The Adam

(Kingma and Ba, 2014) optimizer is used for training and a dropout rate (Srivastava et al., 2014) of 0.3 is enforced during training. The tagger was trained for 10 epochs.

For the lemmatizer, we use a 2-layer biLSTM encoder and a 1-layer LSTM decoder with 400 hidden units. The dimensions of character and tag embedding are 200 and 40, respectively. We enforce a dropout rate of 0.4 in the embedding and encoder LSTM layers. The lemmatizer is also trained with Adam and the learning rate is 0.001. We halve the learning rate whenever the development log-likelihood increases and we perform early-stopping when the learning rate reaches 1×10^{-5}. We apply gradient clipping with a maximum gradient norm of 5.

4.3 Baselines (and Related Work)

We compare our approach against recent competing methods that report results on UD datasets.

Lematus. The current state of the art is held by Bergmanis and Goldwater (2018), who, as discussed in §1, provide a direct context-to-lemma approach, avoiding the use of morphological tags. We remark that Bergmanis and Goldwater (2018) assume a setting where lemmata are annotated at the token level, but morphological tags are not available; we contend, however, that such a setting is not entirely realistic as almost all corpora annotated with lemmata at the token level include morpho-syntactic annotation, including the vast majority of the UD corpora. Thus, we do not consider it a stretch to assume the annotation of morphological tags to train our joint model.[6]

UDPipe. Our next baseline is the UDPipe system of Straka and Straková (2017). Their system performs lemmatization using an averaged perceptron tagger that predicts a (lemma rule, UPOS) pair. Here, a lemma rule generates a lemma by removing parts of the word prefix/suffix and prepending and appending a new prefix/suffix. A guesser first produces correct lemma rules and the tagger is used to disambiguate from them.

[6] After correspondence with Toms Bergmanis, we would like to clarify this point. While Bergmanis and Goldwater (2018) explores the model in a token-annotated setting, as do we, the authors argue that such a model is better for a very low-resource scenario where the entire sentence is not annotated for lemmata. We concede this point—our current model is not applicable in such a setting. However, we note that a semi-supervised morphological tagger could be trained in such a situation as well, which may benefit lemmatization.

Lemming. The strongest non-neural baseline we consider is the system of Müller et al. (2015), who, like us, develop a joint model of morphological tagging lemmatization. In contrast to us, however, their model is globally normalized (Lafferty et al., 2001). Due to their global normalization, they directly estimate the parameters of their model with MLE without worrying about exposure bias. However, in order to efficiently normalize the model, they heuristically limit the set of possible lemmata through the use of **edit trees** (Chrupała, 2008), which makes the computation of the partition function tractable.

Morfette. Much like Müller et al. (2015), Morfette relies on the concept of edit trees. However, a simple perceptron is used for classification with hand-crafted features. A full description of the model is given in Chrupała et al. (2008).

5 Results and Discussion

Experimentally, we aim to show three points. i) Our joint model (eq. (1)) of morphological tagging and lemmatization achieves state-of-the-art accuracy; this builds on the findings of Bergmanis and Goldwater (2018), who show that context significantly helps neural lemmatization. Moreover, the upper bound for contextual lemmatizers that make use of morphological tags is much higher, indicating room for improved lemmatization with better morphological taggers. ii) We discuss a number of error patterns that the model seems to make on the languages, where absolute accuracy is lowest: Latvian, Estonian and Arabic. We suggest possible paths forward to improve performance. iii) We offer an explanation for *when* our joint model does better than the context-to-lemma baseline. We show through a correlational study that our joint approach with morphological tagging helps the most in two cases: low-resource languages and morphologically rich languages.

5.1 Main Results

The first experiment we run focuses on pure performance of the model. Our goal is to determine whether joint morphological tagging and lemmatization improves average performance in a state-of-the-art neural model.

Evaluation Metrics. For measuring lemmatization performance, we measure the accuracy of guessing the lemmata correctly over an entire

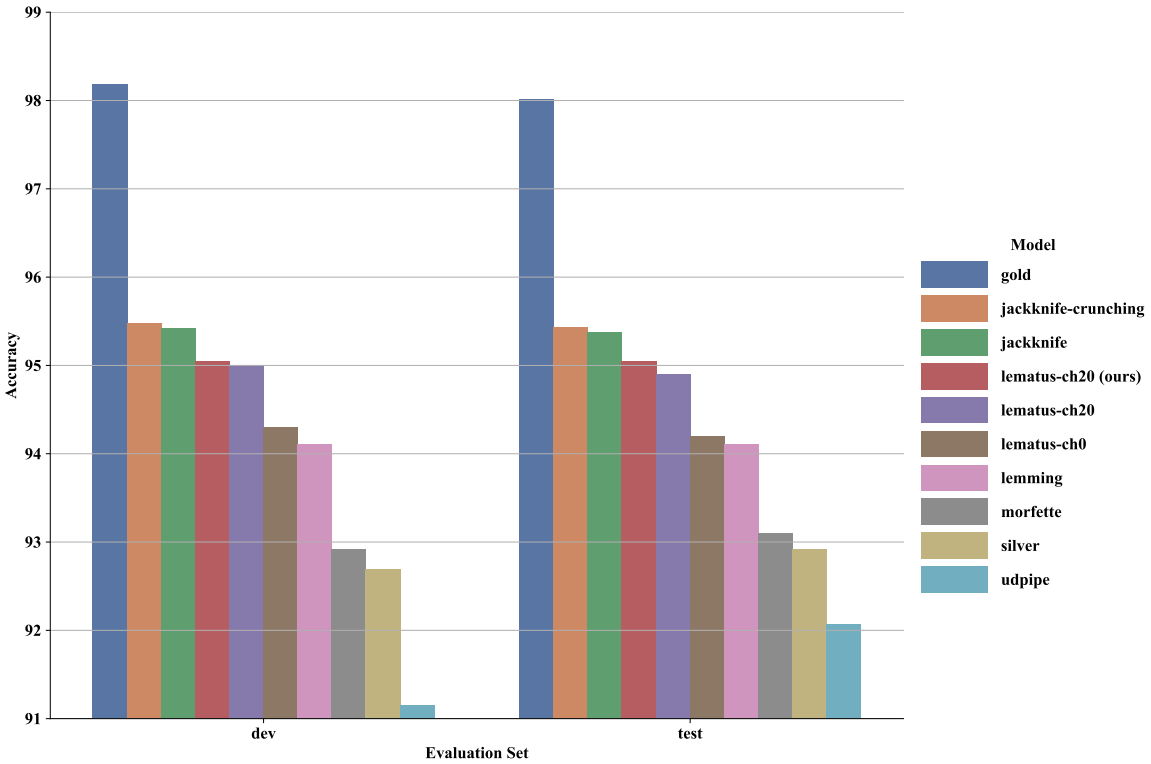

Figure 3: We present performance (in accuracy) averaged over the 20 languages from UD we consider. Our method (second from the left) significantly outperforms the strongest baseline (fourth from the left; Bergmanis and Goldwater (2018)). The blue column is a skyline that gives our model gold tags during decoding, showing improved tagging should lead to better lemmatization. The remaining are baselines described in §4.3.

corpus. To demonstrate the effectiveness of our model in utilizing context and generalizing to unseen word forms, we follow Bergmanis and Goldwater (2018) and also report accuracies on tokens that are i) **ambiguous**, i.e., more than one lemmata exist for the same inflected form, ii) **unseen**, i.e., where the inflected form has not been seen in the training set, and iii) **seen unambiguous**, i.e., where the inflected form has only one lemma and is seen in the training set.

Results. The results showing comparisons with all other methods are summarized in Figure 3. Each bar represents the average accuracy across 20 languages. Our method achieves an average accuracy of 95.42 and the strongest baseline, Bergmanis and Goldwater (2018), achieves an average accuracy of 95.05. The difference in performance (0.37) is statistically significant with $p < 0.01$ under a paired permutation test. We outperform the strongest baseline in 11 out of 20 languages and underperform in only 3 languages with $p < 0.05$. The difference between our method and all other baselines is statistical significant with $p < 0.001$ in all cases. We highlight two additional features of the data. First, decoding using gold morphological tags gives an accuracy of 98.04 for a difference in performance of +2.62. We take the large difference between the upper bound and the current performance of our model to indicate that improved morphological tagging is likely to significantly help lemmatization. Second, it is noteworthy that training with gold tags, but decoding with predicted tags, yields performance that is significantly worse than every baseline except for UDPipe. This speaks for the importance of jackknifing in the training of joint morphological tagger-lemmatizers that are directed and, therefore, suffer from exposure bias.

In Figure 4, we observed crunching further improves performance of the greedy decoding scheme. In 8 out of 20 languages, the improvement is statistical significant with $p < 0.05$. We select the best k for each language based on the development set.

In Figure 5, we provide a language-wise breakdown of the performance of our model and the model of Bergmanis and Goldwater (2018). Our strongest improvements are seen in Latvian, Greek and Hungarian. When measuring performance solely over unseen inflected forms, we achieve even

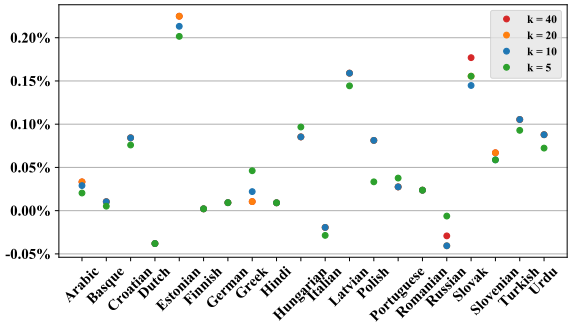

Figure 4: Relative improvement on validation set with crunching over greedy decoding for different values of k.

stronger gains over the baseline method in most languages. This demonstrates the generalization power of our model beyond word forms seen in the training set. In addition, our accuracies on ambiguous tokens are also seen to be higher than the baseline on average, with strong improvements on highly inflected languages such as Latvian and Russian. Finally, on seen unambiguous tokens, we note improvements that are similar across all languages.

5.2 Error Patterns

We attempt to identify systematic error patterns of our model in an effort to motivate future work. For this analysis, we compare predictions of our model and the gold lemmata on three languages with the weakest absolute performance: Estonian, Latvian and Arabic. First, we note the differences in the average lengths of gold lemmata in the tokens we guess incorrectly and all the tokens in the corpus. The lemmata we guess incorrectly are on average 1.04 characters longer than the average length of all the lemmata in the corpus. We found that the length of the incorrect lemmata does not correlate strongly with their frequency. Next, we identify the most common set of edit operations in each language that would transform the incorrect hypothesis to the gold lemma. This set of edit operations was found to follow a power-law distribution.

For the case of Latvian, we find that the operation {*replace*: s $\rightarrow$ a} is the most common error made by our model. This operation corresponds to a possible issue in the Latvian treebank, where adjectives were marked with gendered lemmas. This issue has now been resolved in the latest version of the treebank. For Estonian, the operation {*insert*: m, *insert*: a} is the most common error. The suffix *-ma* in Estonian is used to indicate the infinitive

lang	# tokens	# tags	ours	Lematus	Δ
arabic	202000	349	93.1	93.55	-0.48
basque	59700	884	96.74	96.55	0.2
croatian	146000	1105	96.16	95.7	0.48
dutch	163000	62	97.26	97.65	-0.4
estonian	17000	482	85.83	84.99	0.99
finnish	135000	1669	94.79	94.31	0.51
german	227000	683	97.46	97.72	-0.26
greek	36500	346	95.29	94.22	1.13
hindi	261000	939	98.88	98.92	-0.05
hungarian	16700	580	96.13	94.99	1.2
italian	236000	278	97.93	98.04	-0.11
latvian	28000	640	88.67	87.31	1.56
polish	52000	991	95.99	95.12	0.91
portuguese	176000	375	98.2	98.26	-0.06
romanian	157000	451	97.11	97.19	-0.08
russian	58400	715	96.0	95.07	0.98
slovak	64800	1186	93.25	92.43	0.89
slovenian	96500	1101	97.07	96.9	0.17
turkish	31200	972	95.81	95.01	0.85
urdu	101000	1001	96.76	97.12	-0.37

Table 1: Here we present the number of tokens in each of the UD treebanks we use as well as the number of morphological tags. Note, we take the number of tags as a proxy for the morphological complexity of the language. Finally, we present numbers on validation set from our method with greedy decoding and from the strongest baseline (Lematus) as well as the difference. Correlations between the first two columns and the differences are shown in Table 2.

form of verbs. Gold lemmata for verbs in Estonian are marked in their infinitive forms whereas our system predicts the stems of these verbs instead. These inflected forms are usually ambiguous and we believe that the model doesn't generalize well to different form-lemma pairs, partly due to fewer training data available for Estonian. This is an example of an error pattern that could be corrected using improved morphological information about the tokens. Finally, in Arabic, we find that the most common error pattern corresponds to a single ambiguous word form, *'an* , which can be lemmatized as *'anna* (like "that" in English) or *'an* (like "to" in English) depending on the usage of the word in context. The word *'anna* must be followed by a nominal sentence while *'an* is followed by a verb. Hence, models that can incorporate rich contextual information would be able to avoid such errors.

5.3 Why our model performs better?

Simply presenting improved results does not entirely satiate our curiosity: we would also like to understand *why* our model performs better. Specifically, we have assumed an additional level of

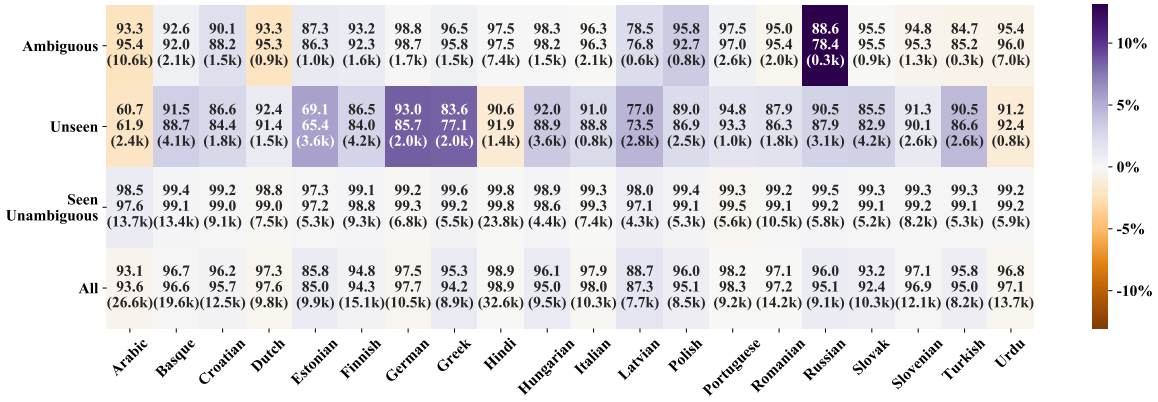

Figure 5: Dev accuracy breakdown by type of inflected form on all languages comparing our system with greedy decoding against our run of Lematus-ch20, colored by relative improvement in percentage. In each entry, the bottom score is from Lematus-ch20 and the top one is from our system, and the number in the parenthesis is the number of tokens for the corresponding setting.

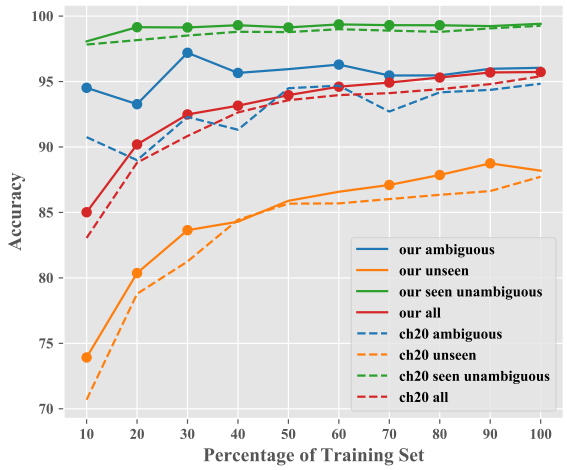

Figure 6: Learning curve showing the accuracy on the validation set of the Polish treebank as the percentage of training set is increased. Markers indicate statistical significant better system with paired permutation test ($p < 0.05$). Our model is decoded greedily.

supervision—namely, the annotation of morphological tags. We provide the differences between our method and our retraining of the Lematus system presented in Table 1. In addition to the performance of the systems, we also list the number of tokens in each treebank and the number of distinct morphological tags per language. We perform a correlational study, which is shown in Table 2.

Morphological Complexity and Performance. We see that there is a moderate positive correlation ($\rho = 0.209$) between the number of morphological tags in a language and the improvement our model obtains. As we take the number of tags as a proxy for the morphological complexity in the language,

	Pearson's Rv	Spearman's ρ
# tags vs. Δ	0.206	0.209
# tokens vs. Δ	-0.808	-0.845

Table 2: The table shows the correlations between the differences in dev performance between our model with greedy decoding and Lematus and two aspects of the data: number of tokens and number of tags.

we view this as an indication that attempting to directly extract the relevant morpho-syntactic information from the corpus is not as effective when there is more to learn. In such languages, we recommend exploiting the additional annotation to achieve better results.

Amount of Data and Performance. The second correlation we find is a stronger negative correlation ($\rho = -0.845$) between the number of tokens available for training in the treebank and the gains in performance of our model over the baseline. This is further demonstrated by the learning curve plot in Figure 6, where we plot the validation accuracy on the Polish treebank for different sizes of the training set. The gap between the performance of our model and Lematus-ch20 is larger when fewer training data are available, especially for ambiguous tokens. This indicates that the incorporation of morphological tags into a model helps more in the low-resource setting. Indeed, this conclusion makes sense—neural networks are good at extracting features from text when there is a sufficiently large amount of data. However, in the low-resource case, we would expect direct super-

vision on the sort of features we desire to extract to work better. Thus, our second recommendation is to model tags jointly with lemmata when fewer training tokens are available. As we noted earlier, it is almost always the case that token-level annotation of lemmata comes with token-level annotation of morphological tags. In low-resource scenarios, a data augmentation approach such as the one proposed by Bergmanis and Goldwater (2019) can be helpful and serve complementary to our approach.

6 Conclusion

We have presented a simple joint model for morphological tagging and lemmatization and discussed techniques for training and decoding. Empirically, we have shown that our model achieves state-of-the-art results, hinting that explicitly modeling morphological tags is a more effective manner for modeling context. In addition to strong numbers, we tried to explain *when* and *why* our model does better. Specifically, we show a significant correlation between our scores and the number of tokens and tags present in a treebank. We take this to indicate that our method improves performance more for low-resource languages as well as morphologically rich languages.

Acknowledgments

We thank Toms Bergmanis for his detailed feedback on the accepted version of the manuscript. Additionally, we would like to thank the three anonymous reviewers for their valuable suggestions. The last author would like to acknowledge support from a Facebook Fellowship.

References

Željko Agić and Natalie Schluter. 2017. How (not) to train a dependency parser: The curious case of jackknifing part-of-speech taggers. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 679–684. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Toms Bergmanis and Sharon Goldwater. 2018. Context sensitive neural lemmatization with Lematus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1391–1400. Association for Computational Linguistics.

Toms Bergmanis and Sharon Goldwater. 2019. Data Augmentation for Context-Sensitive Neural Lemmatization Using Inflection Tables and Raw Text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 33–36. Association for Computational Linguistics.

Victor Chahuneau, Eva Schlinger, Noah A. Smith, and Chris Dyer. 2013. Translating into morphologically rich languages with synthetic phrases. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1677–1687.

Grzegorz Chrupała. 2008. *Towards a machine-learning architecture for lexical functional grammar parsing*. Ph.D. thesis, Dublin City University.

Grzegorz Chrupała, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with Morfette. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA). Http://www.lrec-conf.org/proceedings/lrec2008/.

Ryan Cotterell and Georg Heigold. 2017. Cross-lingual character-level neural morphological tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 748–759. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sebastian Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. The CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 1–27. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*,

pages 1–30. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22. Association for Computational Linguistics.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Alexander Fraser, Marion Weller, Aoife Cahill, and Fabienne Cap. 2012. Modeling inflection and word-formation in smt. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 664–674. Association for Computational Linguistics.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

Martin Haspelmath and Andrea Sims. 2013. *Understanding morphology*. Routledge.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2017. An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 505–513. Association for Computational Linguistics.

Jakub Kanis and Lucie Skorkovská. 2010. Comparison of different lemmatization approaches through the means of information retrieval performance. In *International Conference on Text, Speech and Dialogue*, pages 93–100. Springer.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: Principles and techniques*. MIT Press.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA.

Peter Makarov and Simon Clematide. 2018. UZH at CoNLL-SIGMORPHON 2018 shared task on universal morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 69–75. Association for Computational Linguistics.

Chaitanya Malaviya, Matthew R. Gormley, and Graham Neubig. 2018. Neural factor graph models for cross-lingual morphological tagging. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2653–2663. Association for Computational Linguistics.

Jonathan May and Kevin Knight. 2006. A better n-best list: Practical determinization of weighted finite tree automata. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*.

Arya D McCarthy, Miikka Silfverberg, Ryan Cotterell, Mans Hulden, and David Yarowsky. 2018. Marrying universal dependencies and universal morphology. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 91–101.

Christof Monz and Maarten De Rijke. 2001. Shallow morphological analysis in monolingual information retrieval for Dutch, German, and Italian. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 262–277. Springer.

Thomas Müller, Ryan Cotterell, Alexander 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*, pages 2268–2274. Association for Computational Linguistics.

Joakim Nivre et al. 2017. Universal dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 623–633. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 3104–3112.

John Sylak-Glassman. 2016. The composition and use of the universal morphological feature schema (unimorph schema). Technical report.

Shijie Wu, Pamela Shapiro, and Ryan Cotterell. 2018. Hard non-monotonic attention for character-level transduction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4425–4438. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. Conll 2018 shared task: Multilingual parsing from raw text to universal dependencies. *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21.

A Additional Results

We present the exact numbers on all languages to allow future papers to compare to our results in Table 3 and Table 4.

	Gold	Crunching	Jackknifing	Ch-20	Silver
Arabic	97.40	93.13	93.10	93.55	89.32
Basque	98.70	96.75	96.74	96.55	94.88
Croatian	98.61	96.24	96.16	95.70	94.89
Dutch	98.82	97.26	97.26	97.65	96.31
Estonian	91.92	86.02	85.83	84.99	71.65
Finnish	97.01	94.79	94.79	94.31	90.89
German	97.74	97.47	97.46	97.72	96.27
Greek	97.40	95.33	95.29	94.22	94.33
Hindi	99.29	98.89	98.88	98.92	98.63
Hungarian	98.54	96.22	96.13	94.99	94.15
Italian	99.37	97.93	97.93	98.04	96.95
Latvian	97.24	88.81	88.67	87.31	85.58
Polish	98.46	96.07	95.99	95.12	91.77
Portuguese	99.63	98.24	98.20	98.26	97.72
Romanian	99.34	97.13	97.11	97.19	95.99
Russian	98.46	96.00	96.00	95.07	93.84
Slovak	97.14	93.41	93.25	92.43	88.49
Slovenian	99.46	97.13	97.07	96.90	95.64
Turkish	98.71	95.91	95.81	95.01	91.25
Urdu	97.48	96.84	96.76	97.12	96.62
AVERAGE	98.04	95.48	95.42	95.05	92.76

Table 3: Development performance breakdown.

	Gold	Crunching	Jackknifing	Ch-20	Silver
Arabic	97.95	93.99	93.92	94.16	91.37
Basque	98.54	96.63	96.67	96.49	94.57
Croatian	98.24	95.63	95.58	95.22	94.28
Dutch	98.43	97.25	97.25	97.21	96.50
Estonian	92.34	86.33	86.13	85.44	73.41
Finnish	97.02	94.34	94.29	93.94	90.57
German	97.39	97.14	97.07	97.63	95.88
Greek	97.83	96.53	96.46	95.32	95.05
Hindi	99.10	98.68	98.65	98.73	98.47
Hungarian	97.72	94.02	93.96	93.15	92.42
Italian	99.33	97.83	97.83	98.05	96.96
Latvian	96.69	89.79	89.76	88.87	86.49
Polish	98.45	95.74	95.78	94.90	91.94
Portuguese	99.60	97.97	97.86	98.14	97.58
Romanian	99.55	97.21	97.14	97.21	96.36
Russian	98.30	95.82	95.82	94.77	93.71
Slovak	97.40	93.46	93.31	92.29	88.63
Slovenian	99.25	96.74	96.66	96.69	95.42
Turkish	99.18	96.48	96.32	95.99	92.14
Urdu	97.87	96.94	96.91	96.77	96.73
AVERAGE	98.01	95.43	95.37	95.05	92.92

Table 4: Test performance breakdown.

	F1 Score
Arabic	85.62
Basque	83.68
Croatian	85.37
Dutch	90.92
Estonian	65.80
Finnish	87.94
German	79.45
Greek	87.63
Hindi	87.89
Hungarian	86.00
Italian	93.78
Latvian	80.96
Polish	80.29
Portuguese	93.65
Romanian	93.51
Russian	83.69
Slovak	64.53
Slovenian	88.81
Turkish	82.60
Urdu	72.86
AVERAGE	83.75

Table 5: Morphological Tagging Performance on development set.