Context Sensitive Neural Lemmatization with Lematus

Toms Bergmanis
School of Informatics
University of Edinburgh
T.Bergmanis@sms.ed.ac.uk

Sharon Goldwater
School of Informatics
University of Edinburgh
sgwater@inf.ed.ac.uk

Abstract

The main motivation for developing context-sensitive lemmatizers is to improve performance on unseen and ambiguous words. Yet previous systems have not carefully evaluated whether the use of context actually helps in these cases. We introduce Lematus, a lemmatizer based on a standard encoder-decoder architecture, which incorporates character-level sentence context. We evaluate its lemmatization accuracy across 20 languages in both a full data setting and a lower-resource setting with 10k training examples in each language. In both settings, we show that including context significantly improves results against a context-free version of the model. Context helps more for ambiguous words than for unseen words, though the latter has a greater effect on overall performance differences between languages. We also compare to three previous context-sensitive lemmatization systems, which all use pre-extracted edit trees as well as hand-selected features and/or additional sources of information such as tagged training data. Without using any of these, our context-sensitive model outperforms the best competitor system (Lemming) in the full-data setting, and performs on par in the lower-resource setting.

1 Introduction

Lemmatization is the process of determining the dictionary form of a word (e.g. *swim*) given one of its inflected variants (e.g. *swims*, *swimming*, *swam*, *swum*). Data-driven lemmatizers face two main challenges: first, to generalize beyond the training data in order to lemmatize unseen words; and second, to disambiguate ambiguous wordforms from their sentence context. In Latvian, for example, the wordform "*ceļu*" is ambiguous when considered in isolation: it could be an inflected variant of the verb "*celt*" (*to lift*) or the nouns "*celis*" (*knee*) or "*ceļš*" (*road*); without context, the lemmatizer can only guess.

By definition, sentence context (or latent information derived from it, such as the target word's morphosyntactic tags) is needed in order to correctly lemmatize ambiguous forms such as the example above. Previous researchers have also assumed that context should help in lemmatizing unseen words (Chrupała, 2006; Müller et al., 2015)—i.e., that the context contains useful features above and beyond those in the wordform itself. Nevertheless, we are not aware of any previous work that has attempted to quantify how much (or even whether) context actually helps in both of these cases. Several previous papers on context-sensitive lemmatization have reported results on unseen words (Chrupała, 2006; Chrupała et al., 2008; Müller et al., 2015; Chakrabarty et al., 2017), and some have compared versions of their systems that use context in different ways (Müller et al., 2015; Chakrabarty et al., 2017), but there are few if any direct comparisons between context-sensitive and context-free systems, nor have results been reported on ambiguous forms.

This paper presents *Lematus*—a system that adapts the neural machine translation framework of Sennrich et al. (2017) to learn context sensitive lemmatization using an encoder-decoder model. Context is represented simply using the character contexts of each form to be lemmatized, meaning that our system requires fewer training resources than previous systems: only a corpus with its lemmatized forms, without the need for POS tags (Chrupała et al., 2008; Müller et al., 2015) or word embeddings trained on a much larger corpus (Chakrabarty et al., 2017). We evaluate Lematus on data from 20 typologically varied languages, both using the full training data from the Universal Dependencies project (Nivre et al., 2017), as well as a lower-resource scenario with only 10k training tokens per language. We compare results to three previous systems and to a context-free version of our own system, including results on both unseen and ambiguous words. We also examine the extent to which the rate of unseen and ambiguous words in a language can predict lemmatization performance.

On average across the 20 languages, the context-sensitive version of Lematus achieves significantly higher lemmatization accuracy than its context-free counterpart in both the low-resource and full-data settings. It also outperforms the best competitor system (Lemming; Müller et al. 2015) in the full-data setting, and does as well as Lemming in the low-resource setting. Thus, even without explicitly training on or predicting POS tags, Lematus seems able to implicitly learn similar information from the raw character context.

Analysis of our full-data results shows that including context in the model improves its accuracy more on ambiguous words (from 88.8% to 92.4% on average) than on unseen words (from 83.6% to 84.3% on average). This suggests that, to the extent that unseen words can be correctly lemmatized at all, the wordform itself provides much of the information needed to do so, and Lematus effectively exploits that information—indeed, Lematus without context outperforms all previous context-sensitive models on lemmatizing unseen words.

Finally, our cross-linguistic analysis indicates that the proportions of unseen words and ambiguous words in a language are anti-correlated. Altogether, then, our results suggest that context-free neural lemmatization is surprisingly effective, and may be a reasonable option if the language contains many unseen words but few ambiguous ones. Context is likely to help in most languages, but the main boost is for languages with higher ambiguity.

2 Background and Baseline Systems

Early work on context-sensitive lemmatization focused on disambiguation: given a set of analyses produced by a hand-built morphological analyzer (typically including both lemmas and morphosyntactic tags), choose the best one in context (Oflazer and Kuruöz, 1994; Ezeiza et al., 1998; Hakkani-Tür et al., 2002). Here, we focus on systems learning to generate the lemmas and tags without a pre-existing analyzer (Erjavec and Džeroski, 2004; Chrupała, 2006). The three systems we use as baselines follow Chrupała (2006) in treating the task as a classification problem, where the system learns to choose which of a set of *edit scripts* or *edit trees* (previously induced from the aligned wordform-lemma pairs) should be applied to transform each wordform into the correct lemma.

Two of our baselines, **Morfette**[1] (Chrupała et al., 2008) and **Lemming**[2] (Müller et al., 2015), learn from morphologically annotated corpora to jointly tag each word and lemmatize it by choosing an edit script. Morfette consists of two log-linear classifiers—one for lemmatization and one for tagging—which are combined using beam search to find the best sequence of lemma-tag pairs for all words in the input sentence. Lemming (which proves to be the strongest baseline) also consists of two log-linear components (a classifier for lemmatization and a sequence model for tagging), which are combined either using a pipeline (first tag, then lemmatize) or through joint inference. The lemmatization model uses a variety of features from the edit trees, alignments, orthography of the lemma, and morphosyntactic tags.

In experiments on six languages, Müller et al. (2015) showed that the joint Lemming model worked better than the pipelined model, and that adding morphosyntactic features helped. They also demonstrated improvements over an earlier context-free baseline model (Jiampojamarn et al., 2008). However, they did not evaluate on ambiguous forms, nor directly compare context-sensitive and context-free versions of their own model.

Our third baseline, **Ch-2017**[3] (Chakrabarty et al., 2017) uses a neural network rather than a log-linear model, but still treats lemmatization as a classification task to choose the correct edit tree. (Like our model, Ch-2017 does not perform morphological tagging.) The model composes syntactic and semantic information using two successive bidirectional GRU networks. The first bidirectional GRU network is similar to the character to word model by Ling et al. (2015) and learns syntactic information. The semantic information comes from word embeddings pre-trained on much larger corpora. The second GRU uses a composition of the semantic and syntactic embeddings for the edit tree classification task.

Rather than treating lemmatization as classification, our own model is inspired by recent work on morphological reinflection. As defined by two recent Shared Tasks (Cotterell et al., 2016, 2017), a morphological reinflection system gets as input

[1] https://sites.google.com/site/morfetteweb/

[2] http://cistern.cis.lmu.de/lemming

[3] https://github.com/onkarpandit00786/neural-lemmatizer

some inflected wordform (and possibly its morphosyntactic tags) along with a set of target tags. The system must produce the correct inflected form for the target tags. In the 2016 SIGMORPHON Shared Task, various neural sequence-to-sequence models gave the best results (Aharoni et al., 2016; Kann and Schütze, 2016; Östling, 2016). We base our work closely on one of these (Kann and Schütze, 2016), which also won one of the 2017 tasks (Bergmanis et al., 2017). Our lemmatization task can be viewed as a specific type of reinflection, but instead of assuming that tags are given in the input (or that the system simply has to guess the tags from the wordform itself, as in some of the Shared Tasks), we investigate whether the information available from the tags can instead be inferred from sentence context.

3 Model Description

Our model is based on the network architecture proposed by Sennrich et al. (2017), which implements an attentional encoder-decoder architecture similar to that of Bahdanau et al. (2015). Namely, our model is a deep attentional encoder-decoder with a 2-layer bidirectional encoder with a gated recurrent unit (GRU) (Cho et al., 2014) and a 2-layer decoder with a conditional GRU (Sennrich et al., 2017) in the first layer followed by a GRU in the second layer. For more architectural details see (Sennrich et al., 2017).

A default implementation of this architecture is available in the *Nematus* toolkit,[4] which we used as our starting point. However, Sennrich et al. (2017) used their model for machine translation, while we work on lemmatization. Since our task is closer to the problem of morphological reinflection described above, we changed some of the default model parameters to follow those used in systems that performed well in the 2016 and 2017 SIGMORPHON Shared Tasks (Kann and Schütze, 2016; Bergmanis et al., 2017). Specifically, we reduced the number of hidden units to 100 and the encoder and decoder embedding size to 300.

The input sequence is a space-separated character representation of a word in its N-character left and right sentence context. For example, with $N = 15$, the Latvian word *ceļu* (the genitive plural of the noun *ceļš*, meaning *road*) could be input as:

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

where `<s>`, `<lc>`, `<rc>` stand for word boundary, left and right context markers respectively. The target output is a sequence of characters forming the lemma of the word: `c e ļ š`

4 Datasets

We contend that the difficulty of the lemmatization task largely depends on three factors: morphological productivity, lexical ambiguity and morphological regularity. One aim of our work is to investigate the extent to which it is possible to predict lemmatization performance for a particular language by operationalizing and measuring these properties. Therefore in this section we provide statistics and some analysis of the datasets used in our experiments. We use the standard splits of the Universal Dependency Treebank (UDT) v2.0[5] (Nivre et al., 2017) datasets for 20 languages: Arabic, Basque, Croatian, Dutch[6], Estonian, Finnish, German, Greek, Hindi, Hungarian, Italian, Latvian, Polish, Portuguese, Romanian, Russian, Slovak, Slovene, Turkish and Urdu. See Figure 1 for training and development data sizes.

Because the amount of training data varies widely between languages, we perform some of our language analysis (and later, system evaluation) on a subset of the data, where we use only the first 10k tokens in each language for training. The 10k setting provides a clearer comparison between languages in terms of their productivity, ambiguity, and regularity, and also gives a sense of how much training data is needed to achieve good performance.

One of the main purposes of data-driven lemmatization is to handle unseen words at test time, yet languages with differing morphological productivity will have very different proportions of unseen words. Figure 2 shows the percentage of tokens in the development sets of each language that are not seen in training. Two conditions are given: the full training/development sets, and train/dev sets that are controlled in size across languages. For

[4] https://github.com/EdinburghNLP/nematus

[5] UTD v2.0 datasets are archived at http://hdl.handle.net/11234/1-1983

[6] We use UDT v2.1 dataset for Dutch due to inconsistencies in v2.0.

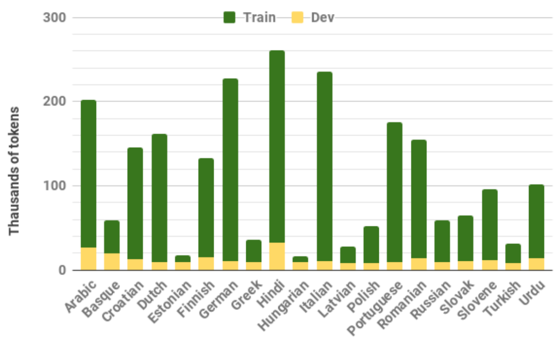


Figure 1: Training and development set sizes for each language, in thousands.

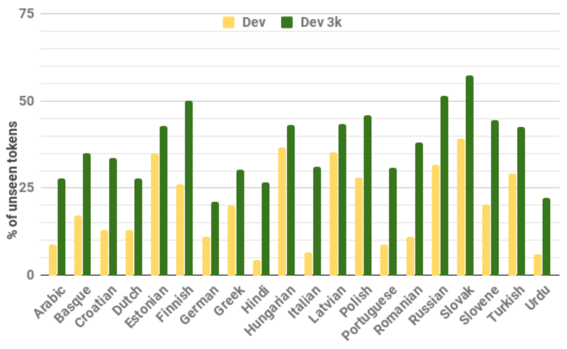


Figure 2: Percent of tokens unseen in training. Dev (yellow): within full development sets with respect to the full training sets. Dev 3k (green): within the first 3k tokens of development sets with respect to the first 10k tokens of training sets.

the languages with large data sets, the percentage of unseen words is (unsurprisingly) higher when training data is reduced to 10k. However, these differences are often small compared to the differences between languages, suggesting that productivity is likely to affect lemmatization performance as much as training data size.

Lexical ambiguity is the other major motivation for context-sensitive lemmatization. To quantify how frequently lemmatizers have to rely on context, Figure 3 shows the percentage of ambiguous tokens in each language, in either the full or reduced training sets. We define ambiguity empirically: ambiguous tokens are wordforms occurring with more than one lemma within the training set.

Overall, the level of measured ambiguity tends to be lower than the proportion of unseen tokens. Many of the languages with high productivity (e.g., Russian, Slovak, Slovene, Turkish) have low levels of ambiguity, while others (Arabic, Urdu) trend the opposite way. Indeed, across all 20 languages,

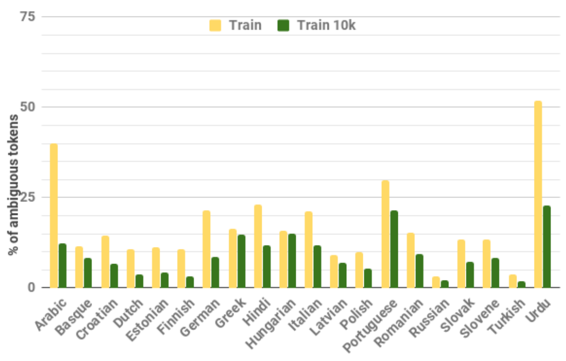


Figure 3: Percent of ambiguous tokens within the first 10k tokens of training sets and full training sets. Ambiguous tokens are word forms occurring with more than one lemma in the training set.

the levels of productivity and ambiguity are negatively correlated, with a rank correlation of -0.57 after controlling for training data size.[7] This is not surprising, since given a set of morphosyntactic functions, they must either be expressed using distinct forms (leading to higher productivity) or non-distinct forms (leading to higher ambiguity).

The final characteristic that we would expect to make some languages easier than others is morphological regularity, but it is unclear how to measure this property directly without an in-depth understanding of the morphophonological rules of a language. Nevertheless, the presence of many irregular forms, or other phenomena such as vowel harmony or spelling changes, complicates lemmatization and will likely affect accuracy.

5 Experimental Setup

Training Parameters[8] We use a mini batch size of 60 and a maximum sequence length of 75. For training we use stochastic gradient descent, Adadelta (Zeiler, 2012), with a gradient clipping threshold of 1.0, recurrent Bayesian dropout probability 0.2 (Gal and Ghahramani, 2016) and weight normalization (Salimans and Kingma, 2016). We use early stopping with patience 10 (Prechelt, 1998). We use the first 10 epochs as a burn-in period, after which at the end of every second epoch

[7]That is, the correlation is computed between the values in Figure 2 Dev 3k (unseen words wrt the first 10k training tokens for each language) and Figure 3 Train 10k (ambiguous words in the first 10k training tokens for each language). The correlation is significantly different from zero with $p < 0.01$.

[8]Training parameters were tunned/verified on the standard splits of UDT training and development sets for Spanish and Catalan, therefore the results on these languages are not included in our evaluation.

we evaluate the current model's lemmatization exact match accuracy on the development set and keep this model if it performs better than the previous best model. When making predictions we use beam-search decoding with a beam of size 12.

Baselines To train models we use the default settings for Morfette and Lemming. Ch-2017 requires word embeddings, for which we use *fastText*[9] (Bojanowski et al., 2017). For Ch-2017 we set the number of training epochs to 100 and implement early stopping with patience 10.[10] We leave the remaining model parameters as suggested by Chakrabarty et al. (2017).

We also use a lookup-based baseline (**Baseline**). For words that have been observed in training, it outputs the most frequent lemma (or the first observed lemma, if the options are equally frequent). For unseen words it outputs the wordform itself as the hypothesized lemma.

Context Representation We aim to use a context representation that works well across multiple languages, rather than to tune the context individually to each language. In preliminary experiments, we explored several different context representations: words, sub-word units, and N surrounding characters, for different values of N. These experiments were carried out on only six languages. Three of these (Latvian, Polish and Turkish) were also used in our main experiments, while three (Bulgarian, Hebrew, and Persian) were not, due to problems getting all the baseline systems to run on those languages.

For the word level context representation (**Words**), we use all words in the left and the right sentence contexts. For the character level context representations (**N-Ch**) we experiment with $N = 0, 5, 10, 15, 20$, or 25 characters of left and right contexts. For the sub-word unit context representation, we use byte pair encoding (**BPE**) (Gage, 1994), which has shown good results for neural machine translation (Sennrich et al., 2016). BPE is a data compression algorithm that iteratively replaces the most frequent pair of symbols (here, characters) in a sequence with a single new symbol. BPE has a single parameter—the number of merge operations. Suitable values for this parameter depend on the application and vary from 10k in language modeling (Vania and Lopez, 2017) to 50k in machine translation (Sennrich et al., 2016). We aim to use BPE to extract a few salient and frequently occurring strings, such as affixes, therefore we set the number of BPE merge operations to 500. We use BPE-encoded left and right sentence contexts that amount up to 20 characters of the original text.

Since we hoped to use context to help with ambiguous words, we looked specifically at ambiguous word performance in choosing the best context representation.[11] Table 1 summarizes Lematus' performance on ambiguous tokens using different sentence context representations. There is no context representation that works best for all six languages, but the 20-Ch system seems to work reasonably well in all cases, and the best on average. We therefore use the 20-Ch context in our main experiments.

Note that this choice was based on a relatively small number of experiments and it is quite possible that further tuning the BPE parameter, or the number of BPE units or words of context (or tuning separately for each language) could lead to better overall results.

Evaluation To evaluate models, we use test and development set lemmatization exact match accuracy. When calculating lemmatization accuracy we ignore casing of the tokens and ommit punctuation tokens and those tokens that contain digits or any of the following characters: `@+._/`.

6 Results and Discussion

Results on Complete Datasets Development set accuracies for all languages and systems in the full data setting are provided in Figure 4a, with results on unseen and ambiguous words in Figures 4b and 4c. Overall, Lematus 20-Ch outperforms the previous systems, Morfette, Lemming and Ch-2017, on 20, 15 and 20 languages respectively. In addition, Figure 4 makes it clear that the major benefit of all the systems over the baseline is for unseen words: in fact, for ambiguous words, the baseline even outperforms some of the systems in a few languages. Comparing the two versions of Lematus, we can see that Lematus 20-Ch does consistently better

[9] `https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md`

[10] We do so because it is unclear what stopping criterion was used by Chakrabarty et al. (2017) Their suggested default for the number of training epochs is 6, yet the values used in their experiments vary from 15 for Hindi to 80 for Bengali.

[11] The percentage of ambiguous tokens in the training sets of Bulgarian, Hebrew and Persian are 8.4%, 16.6% and 7.6% respectively; for the other languages, see Figure 3.

	Baseline	0-Ch	5-Ch	10-Ch	15-Ch	20-Ch	25-Ch	BPE	Words
Bulgarian	81.1	83.0	79.7	88.9	88.2	**89.2**	88.5	89.2	84.2
Hebrew	**95.3**	95.0	82.5	84.9	86.0	86.3	85.4	84.4	75.5
Latvian	73.8	76.6	70.1	73.2	73.9	**74.8**	71.1	71.6	66.2
Persian	**94.4**	92.5	90.5	91.5	91.0	92.5	92.5	93.0	88.0
Polish	90.6	91.7	84.0	84.0	93.0	**93.6**	83.5	85.6	83.0
Turkish	78.6	80.9	75.9	77.9	85.5	**85.9**	79.3	75.9	73.8
Average	85.6	86.6	80.5	83.4	86.3	**87.1**	83.4	83.3	78.5

Table 1: Lemmatization exact match accuracy on ambiguous tokens of dev sets, for baseline and for Lematus using various context representations: N characters, Byte Pair Encoding units, or words.

	Dev				Test	10k:Dev	10k:Test
	All	Unseen	Ambig	SeenUA	All	All	All
Baseline	85.8	39.6	88.0	**99.2***†‡	86.1	74.4	74.4
Morfette	92.9	75.7	91.4	98.9	93.1	86.8	86.5
Lemming	94.1	81.4	**92.4**†	98.8	94.1	87.5	87.3
Ch-2017	90.8	75.0	90.7	96.2	89.8	80.2	79.0
Lematus 0-Ch	94.3	83.6*	88.8	98.9	94.2	87.1	86.6
Lematus 20-Ch	**95.0***†	**84.3***†	**92.4**†	98.8	**94.9***†	**88.4**†	**87.8**†

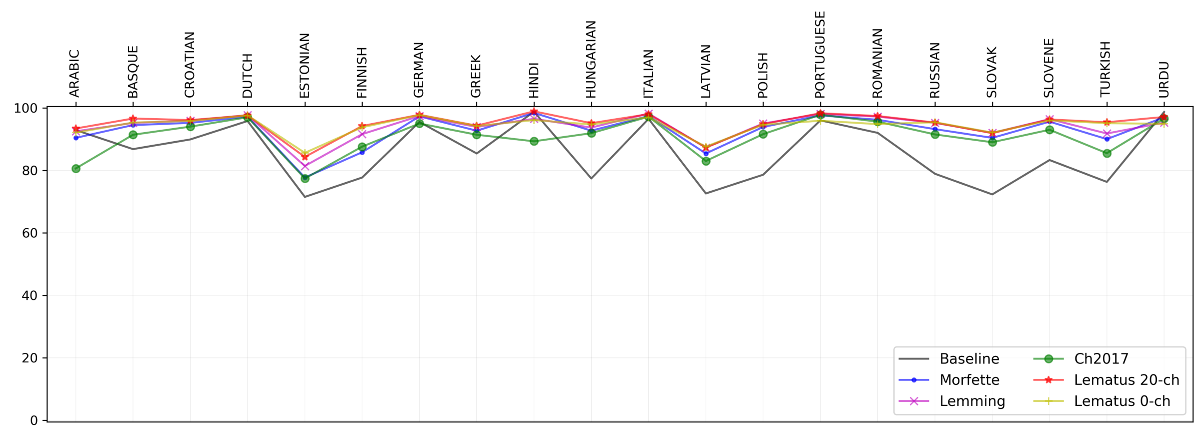
Table 2: Lemmatization exact match accuracy, averaged across all 20 languages. In the full training scenario (first five columns) results are given for All, Unseen, Ambiguous, and Seen Unambiguous tokens. (Note that ambiguity is empirical: is a type seen with more than one lemma in training?) We compare Lematus with/without context (20-Ch/0-Ch), the most frequent lemma baseline, and three previous systems. The numerically highest score in each column is bold; *, †, and ‡ indicate statistically significant improvements over Lemming, Lematus 0-Ch and 20-Ch, respectively (all $p < 0.05$; see text for details).

on ambiguous tokens than Lematus 0-Ch, whereas their performance on unseen tokens (and thus, overall) is much more similar. In fact, on unseen words, Lematus 0-Ch outperforms the context-sensitive baselines Morfette, Lemming and Ch-2017 on 18, 12 and 17 languages respectively. These results suggest that a good context-free model can do surprisingly well on unseen words, and the added model complexity and annotation requirements of earlier context-sensitive models are not always justified.

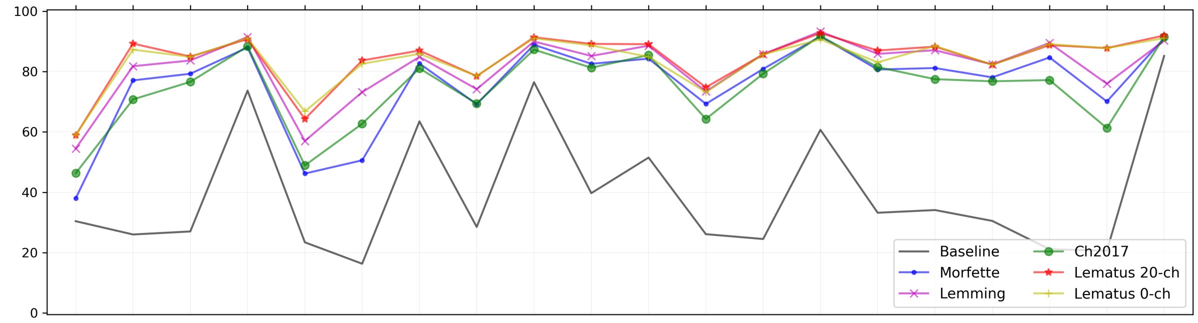
As further evidence of these claims, we summarize in Table 2 each system's average performance over all languages for both the development and test sets. In addition to performance breakdown into unseen and ambiguous words we also report each system's performance on tokens that were both seen and unambiguous in training. No system achieves 100% accuracy on seen unambiguous tokens—even the lookup baseline achieves only 99%, indicating that about 1% of tokens that appeared unambiguous in training occur with a previously unseen lemma in the development set. In principle, context-based systems could outperform the baseline on these words, but in practice none of them do. Indeed, switching to a dictionary lookup baseline for seen unambiguous words would slightly improve the performance of all models (though it would not change the overall ranking of the systems).

We tested for statistically significant differences between the results of Lemming (the numerically best-performing competitor system) and our two systems (Lematus 0-Ch and Lematus 20-Ch) using a Monte Carlo method: for each comparison (say, between 0-Ch and 20-Ch on unseen words), we generated 10000 random samples, where each sample randomly swapped the two systems' results for each language with probability .5. We then obtained a p-value by computing the proportion of samples for which the difference in average results was at least as large as the difference observed in our experiments.

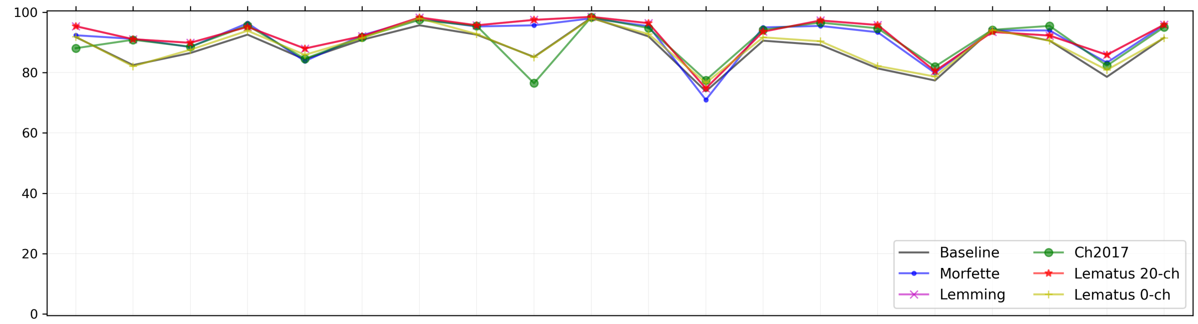
Because the results of 0-Ch and 20-Ch are highly correlated across languages, all differences between these systems, except for results on seen unambiguous tokens, are significant ($p < 0.01$ for dev set All, $p < 0.05$ for Unseen, $p < 0.001$ for Ambig, and $p < 0.01$ for test set All; $p > 0.1$ for



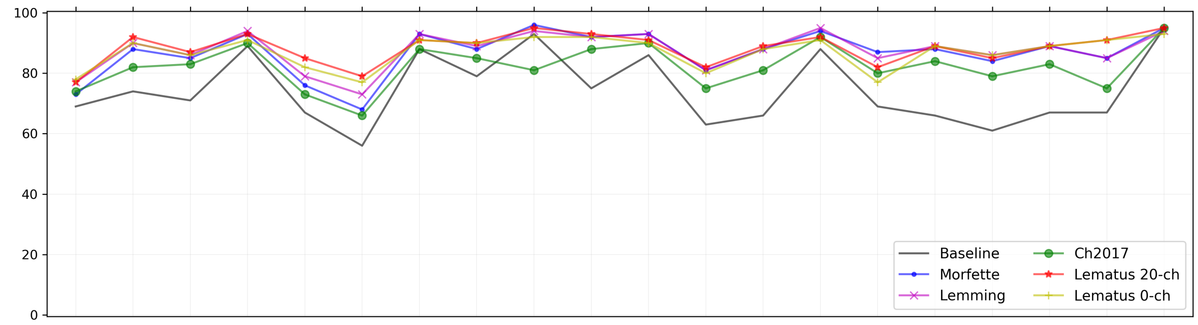
(a) **All tokens.** Models were trained on full training sets.



(b) **Unseen tokens.** Models were trained on full training sets.



(c) **Ambiguous tokens.** Models were trained on full training sets.



(d) **All tokens**. Models were trained on the first **10K** of the training sets.

Figure 4: Lemmatization exact match accuracy on development sets for each language.

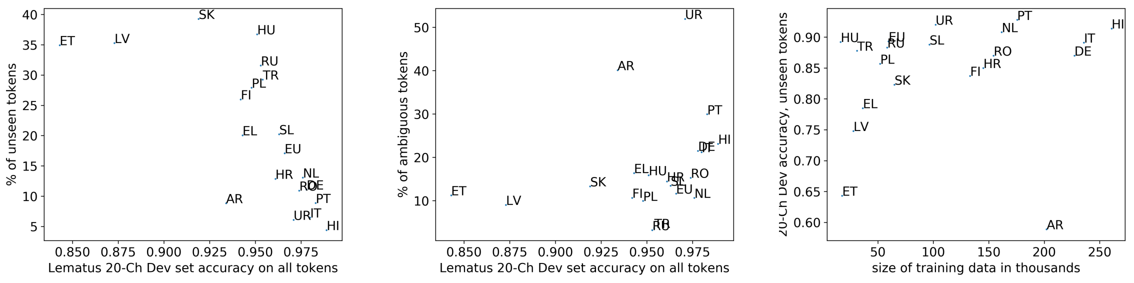


Figure 5: Lemmatization accuracy of Lematus 20-Ch on *all* dev set tokens vs percent of unseen tokens (left) or percent of ambiguous tokens (middle); accuracy on *unseen* tokens vs training set size (right).

dev set SeenUA). Lemming does as well as Lematus 20-Ch on ambiguous and SeenUA words, but its accuracy on unseen words is lower ($p < 0.001$), leading to worse performance overall ($p < 0.01$ on both dev and test). Interestingly, even Lematus 0-Ch does better than Lemming on unseen words ($p < 0.02$), and performs on par overall ($p = 0.28$). So, although including context clearly can help (compare Lematus 20-Ch vs 0-Ch), and Lemming exploits this advantage for ambiguous words, a good context-free model can still do very well. Overall, our models do as well or better than the earlier ones, without the added model complexity and annotation requirements. On the other hand, although our context-sensitive model does improve somewhat over its context-free counterpart, there is still some way to go, since average performance on unseen and ambiguous words is still 84% and 92% respectively.

Results on 10k Datasets Figure 4d shows the results on all tokens for each language in the 10k training setting, with averages in Table 2. On average, limiting training data to the first 10k examples resulted in an 82% reduction of training sets, and we see an average drop in test set performance of 5.6-6.8 percentage points for all systems except Ch-2017, which drops by about 10 percent. When comparing the 0-Ch and 20-Ch versions of Lematus we found the same pattern of significances as in the full data setting ($p < 0.01$), however the two best systems (Lematus 20-Ch and Lemming) are statistically equivalent on the test sets, as are Lemming and Lematus 0-Ch.

Patterns Across Languages In Section 4, we hypothesized that the success of data-driven lemmatization depends on a language's productivity, ambiguity, and regularity. We now explore the extent to which our results support this hypothesis. First, we examine the correlation between the overall performance of our best system on each language and the percentage of unseen (Figure 5, left) or ambiguous words (Figure 5, middle) in that language. As expected, there is a strong negative correlation between the percentage of unseen words and the accuracy of Lematus 20-Ch: the rank correlation is $R = -0.73$ ($p < 0.001$; we use rank correlation because it is less sensitive to outliers than is linear correlation, and the plot clearly shows several outliers.) In contrast to our original prediction, however, Lematus 20-Ch is actually *more* accurate for languages with greater ambiguity ($R = 0.44$, $p = 0.05$). The most likely explanation is that ambiguity is negatively correlated with productivity. Since there tend to be more unseen than ambiguous words, and since accuracy is typically lower for unseen than ambiguous words, higher ambiguity (which implies fewer unseen words) can actually lead to higher overall accuracy.

Our earlier results also suggested that the main benefit of Lematus 20-Ch over Lematus 0-Ch is for ambiguous words. To confirm this, we looked at the extent to which the *difference* in performance between the two systems correlates with the percentage of unseen or ambiguous words in a language. As expected, this analysis suggests that including context in the model helps more for languages with more ambiguity ($R = 0.67, p < 0.001$). In contrast, Lematus 20-Ch provides *less* benefit over Lematus 0-Ch for the languages with more unseen words ($R = -0.75, p < 0.0001$). Again, we assume the latter result is due to the negative correlation between ambiguity and productivity.

So far, our results and analysis show a clear relationship between productivity and ambiguity, and also suggest that using context for lemmatization may be unnecessary (or at least less beneficial) for languages with many unseen words but low ambiguity. However, there are remaining differences between languages that are more difficult to explain. For example, one might expect that for languages with more training data, the system would learn better generalizations and lemmatization accuracy on unseen words would be higher. However, Figure 5 (right), which plots accuracy on *unseen* words in each language as a function of training data size, illustrates that there is no significant correlation between the two variables ($R = 0.32$, $p = 0.16$). In some languages (e.g., Hungarian, in the top left) Lematus performs very well on unseen words even with little training data, while in others (e.g., Arabic, along the bottom) it performs poorly despite relatively large training data. We assume that regularity (and perhaps the nonconcatenative nature of Arabic) must be playing an important role here, but we leave for future work the question of how to operationalize and measure regularity in order to further test this hypothesis.

7 Conclusion

We presented Lematus, a simple sequence-to-sequence neural model for lemmatization that uses character-level context. On average across 20 languages, we showed that even without using context, this model performs as well or better than three previous systems that treated lemmatization as an edit tree classification problem and required POS tags (Chrupała et al., 2008; Müller et al., 2015) or word embeddings trained on a much larger corpus (Chakrabarty et al., 2017). We also showed that with both larger and smaller training datasets, including context boosts performance further by improving accuracy on both unseen and (especially) ambiguous words.

Finally, our analysis suggests that lemmatization accuracy tends to be higher for languages with *low* productivity (as measured by the proportion of unseen words at test time), but more surprisingly also for languages with *high* ambiguity—perhaps because high ambiguity is also associated with low productivity. We also found that the amount of training data available for each language is not a good predictor of performance on unseen words, suggesting that morphological regularity or other language-specific characteristics are playing an important role. Understanding the causes of these differences is likely to be important for further improving neural lemmatization.

8 Acknowledgements

This work was supported in part by the James S McDonnell Foundation (Scholar Award #220020374).

References

Roee Aharoni, Yoav Goldberg, and Yonatan Belinkov. 2016. Improving sequence to sequence learning for morphological inflection generation: The BIU-MIT systems for the SIGMORPHON 2016 shared task for morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. page 41.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training data augmentation for low-resource morphological inflection. In *Proceedings of the CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*. Association for Computational Linguistics, Vancouver, Canada.

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.

Abhisek Chakrabarty, Onkar Arun Pandit, and Utpal Garain. 2017. Context sensitive lemmatization using two successive bidirectional gated recurrent networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1481–1491.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *The Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, Doha, Qatar.

Grzegorz Chrupała. 2006. Simple data-driven context-sensitive lemmatization. *Procesamiento del Lenguaje Natural* 37.

Grzegorz Chrupała, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with morfette. In Bente Maegaard Joseph Mariani Jan Odijk Stelios Piperidis Daniel Tapias Nicoletta Calzolari (Conference Chair), Khalid Choukri, editor, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. European Language Resources Association (ELRA), Marrakech, Morocco.

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*. Association for Computational Linguistics, Vancouver, pages 1–30.

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

Tomaž Erjavec and Sasčo Džeroski. 2004. Machine learning of morphosyntactic structure: Lemmatizing unknown slovene words. *Applied Artificial Intelligence* 18(1):17–41.

Nerea Ezeiza, Iñaki Alegria, José María Arriola, Rubén Urizar, and Itziar Aduriz. 1998. Combining stochastic and rule-based methods for disambiguation in agglutinative languages. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pages 380–384.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.* 12(2):23–38.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*. pages 1019–1027.

Dilek Z. Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2002. Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities* 36(4):381–410.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, Columbus, Ohio, pages 905–913.

Katharina Kann and Hinrich Schütze. 2016. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *Proceedings of ACL*. Association for Computational Linguistics.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1520–1530.

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*. Association for Computational Linguistics, Lisbon, Portugal, pages 2268–2274.

Joakim Nivre et al. 2017. Universal dependencies 2.0 CoNLL 2017 shared task development and test data. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Kemal Oflazer and Ìlker Kuruöz. 1994. Tagging and morphological disambiguation of turkish text. In *Proceedings of the fourth conference on Applied natural language processing*. Association for Computational Linguistics, pages 144–149.

Robert Östling. 2016. Morphological reinflection with convolutional neural networks. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. pages 23–26.

Lutz Prechelt. 1998. Early stopping-but when? *Neural Networks: Tricks of the trade* pages 553–553.

Tim Salimans and Diederik P. Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *CoRR* abs/1602.07868.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. *CoRR* abs/1703.04357.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1715–1725.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2016–2027.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .