Data Augmentation for Context-Sensitive Neural Lemmatization Using Inflection Tables and Raw Text

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

Lemmatization aims to reduce the sparse data problem by relating the inflected forms of a word to its dictionary form. Using context can help, both for unseen and ambiguous words. Yet most context-sensitive approaches require full lemma-annotated sentences for training, which may be scarce or unavailable in low-resource languages. In addition (as shown here), in a low-resource setting a lemmatizer can learn more from n labeled examples of distinct words (types) than from n (contiguous) labeled tokens, since the latter contain far fewer distinct types. To combine the efficiency of type-based learning with the benefits of context, we propose a way to train a context-sensitive lemmatizer with little or no labeled corpus data, using inflection tables from the UniMorph project and raw text examples from Wikipedia that provide sentence contexts for the unambiguous UniMorph examples. Despite these being unambiguous examples, the model successfully generalizes from them, leading to improved results (both overall, and especially on unseen words) in comparison to a baseline that does not use context.

1 Introduction

Many lemmatizers work on isolated wordforms (Wicentowski, 2002; Dreyer et al., 2008; Rastogi et al., 2016; Makarov and Clematide, 2018b,a). Lemmatizing in context can improve accuracy on ambiguous and unseen words (Bergmanis and Goldwater, 2018), but most systems for context-sensitive lemmatization must train on complete sentences labeled with POS and/or morphological tags as well as lemmas, and have only been tested with 20k-300k training tokens (Chrupała et al., 2008; Müller et al., 2015; Chakrabarty et al., 2017).[1]

Intuitively, though, sentence-annotated data is inefficient for training a lemmatizer, especially in low-resource settings. Training on (say) 1000 word types will provide far more information about a language's morphology than training on 1000 contiguous tokens, where fewer types are represented. As noted above, sentence data can help with ambiguous and unseen words, but we show here that when data is scarce, this effect is small relative to the benefit of seeing more word types.[2]

Motivated by this result, we propose a training data augmentation method that combines the efficiency of type-based learning and the expressive power of a context-sensitive model. We use *Lematus* (Bergmanis and Goldwater, 2018), a state-of-the-art lemmatizer that learns from lemma-annotated words in their N-character contexts. No predictions about surrounding words are used, so fully annotated training sentences are not needed. We exploit this fact by combining two sources of training data: 1k lemma-annotated types (with contexts) from the Universal Dependency Treebank (UDT) v2.2[3] (Nivre et al., 2017), plus examples obtained by finding *unambiguous* word-lemma pairs in inflection tables from the Universal Morphology (UM) project[4] and collecting sentence contexts for them from Wikipedia. Although these examples are noisy and biased, we show that they improve lemmatization accuracy in experiments on 10 languages, and that the use of context helps, both overall and especially on unseen words.

2 Method

Lematus[5] (Bergmanis and Goldwater, 2018) is a neural sequence-to-sequence model with attention

[1] The smallest of these corpora contains 20k tokens of Bengali annotated only with lemmas, which Chakrabarty et al. (2017) reported took around two person months to create.

[2] Garrette et al. (2013) found the same for POS tagging.

[3] http://hdl.handle.net/11234/1-2837

[4] http://unimorph.org

[5] https://bitbucket.org/tomsbergmanis/lematus.git

inspired by the re-inflection model of Kann and Schütze (2016), which won the 2016 SIGMORPHON shared task (Cotterell et al., 2016). It uses the architecture of Sennrich et al. (2017): a 2-layer bidirectional GRU encoder and a 2-layer decoder with a conditional GRU (Sennrich et al., 2017) in the first layer and a GRU in the second layer.

Lematus takes as input a character sequence representing the wordform in its N-character context, outputs the characters of the lemma. Special input symbols are used to represent the left and right boundary of the target wordform (`<lc>`, `<rc>`) and other word boundaries (`<s>`). For example, if $N = 15$, the system trained on Latvian would be expected to produce the characters of the lemma *ceļš* (meaning *road*) given input such 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

When $N = 0$ (**Lematus 0-ch**), no context is used, making Lematus 0-ch similar to other systems that do not model context (Dreyer et al., 2008; Rastogi et al., 2016; Makarov and Clematide, 2018b,a). In our experiments we use both Lematus 0-ch and **Lematus 20-ch** (20 characters of context), which was the best-performing system reported by Bergmanis and Goldwater (2018).

2.1 Data Augmentation

Our data augmentation method uses UM inflection tables and creates additional training examples by finding Wikipedia sentences that use the inflected wordforms in context, pairing them with their lemma as shown in the inflection table. However, we cannot use all the words in the tables because some of them are ambiguous: for example, Figure 1 shows that the form `ceļi` could be lemmatized either as `ceļš` or `celis`. Since we don't know which would be correct for any particular Wikipedia example, we only collect examples for forms which are unambiguous according to the UM tables. However, this method is only as good as the coverage of the UM tables. For example, if UM doesn't include a table for the Latvian verb *celt*, then the underlined forms in Table 1 would be incorrectly labeled as unambiguous.

There are several other issues with this method that could potentially limit its usefulness. First, the UM tables only include verbs, nouns and adjectives, whereas we test the system on UDT data, which includes all parts of speech. Second, by excluding

	noun: ceļš SG	noun: ceļš PL	noun: celis SG	noun: celis PL
NOM	ceļš	~~ceļi~~	celis	~~ceļi~~
GEN	~~ceļa~~	~~ceļu~~	~~ceļa~~	~~ceļu~~
DAT	ceļam	~~ceļiem~~	celim	~~ceļiem~~
ACC	ceļu	~~ceļus~~	celi	~~ceļus~~
INS	ceļu	~~ceļiem~~	celi	~~ceļiem~~
LOC	ceļā	~~ceļos~~	celī	~~ceļos~~
VOC	ceļ	~~ceļi~~	celi	~~ceļi~~

Table 1: Example UM inflection tables for Latvian nouns *ceļš* (*road*) and *celis* (*knee*). The ~~crossed out forms~~ are examples of evidently ambiguous forms that are not used for data augmentation because of being shared by the two lemmas. The underlined forms appear unambiguous in this toy example but actually conflict with inflections of the verb *celt* (*to lift*).

ambiguous forms, we may be restricting the added examples to a non-representative subset of the potential inflections, or the system may simply ignore the context because it isn't needed for these examples. Finally, there are some annotation differences between UM and UDT.[6] Despite all of these issues, however, we show below that the added examples and their contexts do actually help.

3 Experimental Setup

Baselines and Training Parameters We use three baselines: (1) **Lemming**[7] (Müller et al., 2015) is a context-sensitive system that uses log-linear models to jointly tag and lemmatize the data, and is trained on sentences annotated with both lemmas and POS tags. (2) The hard monotonic attention model (**HMAM**)[8] (Makarov and Clematide, 2018b) is a neural sequence-to-sequence model with a hard attention mechanism that advances through the sequence monotonically. It is trained on word-lemma pairs (without context) with character-level alignments learned in a preprocessing step using an alignment model, and it has proved to be competitive in low resource scenarios. (3) Our naive **Baseline** outputs the most frequent lemma (or one lemma at random from the options that are equally frequent) for words observed in training. For unseen words it outputs the wordform

[6] Recent efforts to unify the two resources have mostly focused on validating dataset schema (McCarthy et al., 2018), leaving conflicts in word lemmas unresolved. We estimated (by counting types that are unambiguous in each dataset but have different lemmas across them) that annotation inconsistencies affect up to 1% of types in the languages we used.

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

[8] https://github.com/ZurichNLP/coling2018-neural-transition-based-morphology

itself.

To train the models we use the default settings for Lemming and the suggested lemmatization parameters for HMAM. We mainly follow the hyperparameters used by Bergmanis and Goldwater (2018) for Lematus; details are in Appendix A.

Languages and Training Data We conduct preliminary experiments on five development languages: Estonian, Finnish, Latvian, Polish, and Russian. In our final, experiments we also add Bulgarian, Czech, Romanian, Swedish and Turkish. We vary the amount and type of training data (types vs. tokens, UDT only, UM only, or UDT plus up to 10k UM examples), as described in Section 4.

Evaluation To evaluate models' ability to lemmatize wordforms in their sentence context we follow Bergmanis and Goldwater (2018) and use the full UDT development and test sets. Unlike Bergmanis and Goldwater (2018) who reported token level lemmatization exact match accuracy, we report *type-level* micro averaged lemmatization exact match accuracy. This measure better reflects improvements on unseen words, which tend to be rare but are more important (since a most-frequent-lemma baseline does very well on seen words, as shown by Bergmanis and Goldwater (2018)).

We also separately report performance on unseen and ambiguous tokens. For a fair comparison across scenarios with different training sets, we count as unseen only words that are absent from *all* training sets/scenarios and are not ambiguous. Due to the small training sets, between 70-90% of dev set types are classed as unseen in each language. We define a type as ambiguous if the empirical entropy over its lemmas is greater than 0.1 in the full original UDT training splits.[9] According to this measure, only 1.2-5.3% of dev set types are classed as ambiguous in each language.

Significance Testing To test for statistically significant differences between the results of two systems we use a Monte Carlo method: for each set of results we generate 10000 random samples, where each sample swaps the results of the two systems for each language with a probability of 0.5. We then obtain a p-value as the proportion of samples for which the difference on average was at least as large as the difference observed in our experiments.

[9]This measure, *adjusted ambiguity*, was defined by Kirefu (2018), who noticed that many frequent wordforms appear to have multiple lemmas due to annotation errors. The adjusted ambiguity filters out these cases.

		Ambig.	Unseen	All
Tokens	Baseline	41.0	26.6	31.0
	Lemming	38.2	48.3	50.6
	HMAM	41.4	50.2	52.1
	Lematus 0-ch	39.9	43.7	46.8
	Lematus 20-ch	38.4	42.8	45.8
Types	Baseline	43.4	27.4	30.8
	Lemming	N/A	N/A	N/A
	HMAM	47.3	53.9	53.6
	Lematus 0-ch	46.8	52.4	52.6
	Lematus 20-ch	46.9	52.0	52.3

Table 2: Average type level lemmatization exact match accuracy on five development languages in type and token based training data scenarios.

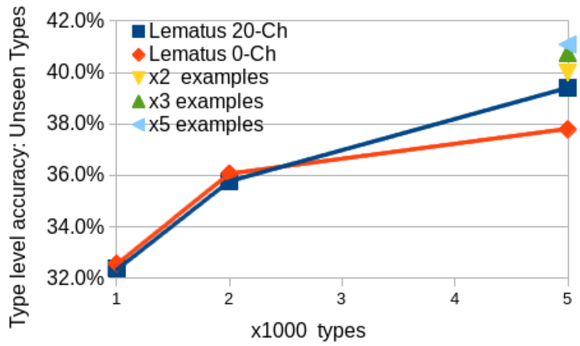

Figure 1: Average type level lemmatization exact match accuracy on unseen words of five development languages. X-axis: thousands of types in training data.

4 Experiments, Results, and Discussion

Types vs. Tokens and Context in Very Low Resource Settings We compare training on the first 1k tokens vs. first 1k distinct types of the UDT training sets. Table 2 shows that if only 1k examples are available, using types is clearly better for all systems. Although Lematus does relatively poorly on the token data, it benefits the most from switching to types, putting it on par with HMAM and suggesting is it likely to benefit more from additional type data. Lemming requires token-based data, but does worse than HMAM (a context-free method) in the token-based setting, and we also see no benefit from context in comparing Lematus 20-ch vs Lematus 0-ch. So overall, in this very low-resource scenario with no data augmentation, context does not appear to help.

		DEVELOPMENT			TEST
		Ambig.	Unseen	All	All
No aug.	*Baseline*	49.1	30.8	36.7	-
	HMAM	46.3	58.9	61.5	61.5
	Lematus 0-ch	46.5	55.0	58.5	59.1
	Lematus 20-ch	45.0	54.3	57.7	57.7
1k UM	*Baseline*	45.9	30.8	38.4	-
	HMAM	45.9	60.2	64.2	64.3
	Lematus 0-ch	46.6	59.0	63.4	63.6
	Lematus 20-ch	49.8	61.7	65.5	65.3
5k UM	*Baseline*	55.3	30.8	41.7	-
	HMAM	46.7	60.8	65.7	65.7
	Lematus 0-ch	46.2	61.6	66.2	66.4
	Lematus 20-ch	48.6	65.4	69.2	69.5
10k UM	*Baseline*	54.9	31.2	43.5	-
	HMAM	45.4	60.8	65.5	65.3
	Lematus 0-ch	46.4	62.8	67.1	66.7
	Lematus 20-ch	49.5	66.7	70.6	70.9

Table 3: Average lemmatization accuracy for all 10 languages, trained on 1k UDT types in context only (No aug.), or 1k UDT plus 1k, 5k, or 10k UM types with contexts from Wikipedia.

Using UM + Wikipedia Only We now try training only on UM + Wikipedia examples, rather than examples from UDT. We use 1k, 2k or 5k unambiguous types from UM with a single example context from Wikipedia for each. With 5k types we also try adding more example contexts (2, 3, or 5 examples for each type).

Figure 1 presents the results (for unseen words only). As with the UDT experiments, there is little difference between Lematus 20-Ch and Lematus 0-Ch in the smallest data setting. However, when the number of training types increases to 5k, the benefits of context begin to show, with Lematus 20-ch yielding a 1.6% statistically significant ($p < 0.001$) improvement over Lematus 0-ch. The results for increasing the number of examples per type are numerically higher than the one-example case, but the differences are not statistically significant.

It is worth noting that the accuracy even with 5k UM types is considerably lower than the accuracy of the model trained on only 1k UDT types (see Table 2). We believe this discrepancy is due to the issues of biased/incomplete data noted above. For example, we analyzed the Latvian data and found that the available tables for nouns, verbs, and adjectives give rise to 78 paradigm slots. The 17 POS tags in UDT give rise to about 10 times as many paradigm slots, although only 448 are present in the unseen words of the dev set. Of these, 197 are represented amongst the 1k UDT training types, whereas only 25 are included in the 1k UM training types. As a result, about 72% of the unseen types of dev set have no representative of their paradigm slot in 1k types of UM, whereas this figure is only 17% for the 1k types of UDT.

Data Augmentation Although UM + Wikipedia examples alone are not sufficient to train a good lemmatizer, they might improve a low-resource baseline trained on UDT data. To see, we augmented the 1k UDT types with 1k, 5k or 10k UM types with contexts from Wikipedia.

Table 3 summarizes the results, showing that despite the lower quality of the UM + Wikipedia examples, using them improves results of all systems, and more so with more examples. Improvements are especially strong for unseen types, which constitute more than 70% of types in the dev set.

Considering the two context-free models, HMAM does better on the un-augmented 1k UDT data, but (as predicted by our results above) it benefits less from data augmentation than does Lematus 0-Ch, so with added data they are statistically equivalent ($p = 0.07$ on the test set with 10k UM).

More importantly, Lematus 20-ch begins to out-

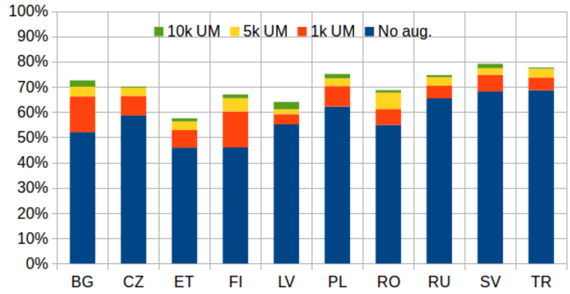

Figure 2: Lematus 20-ch lemmatization accuracy for each language on all types in the dev sets.

perform the context-free models with as few as 1k UM + Wikipedia examples, and the difference increases with more examples, eventually reaching over 4% better on the test set than the next best model (Lematus 0-Ch) when 10k UM + Wikipedia examples are used ($p < 0.001$) This indicates that the system can learn useful contextual cues even from unambiguous training examples.

Finally, Figure 2 gives a breakdown of Lematus 20-ch dev set accuracy for individual languages, showing that data augmentation helps consistently, although results suggest diminishing returns.

5 Conclusion

We proposed a training data augmentation method that combines the efficiency of type-based learning and the expressive power of a context-sensitive lemmatization model. The proposed method uses Wikipedia sentences to provide contextualized examples for unambiguous inflection-lemma pairs from UniMorph tables. These examples are noisy and biased, but nevertheless they improve lemmatization accuracy on all ten languages we tried. In particular, we showed that context is helpful, both overall and especially on unseen words—the first work we know of to demonstrate improvements from context in a very low-resource setting.

References

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.

Toms Bergmanis and Sharon Goldwater. 2018. Context sensitive neural lemmatization with lematus. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

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)*, pages 1481–1491, Vancouver, Canada. Association for Computational Linguistics.

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).

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.

Markus Dreyer, Jason R Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1080–1089. Association for Computational Linguistics.

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-world semi-supervised learning of pos-taggers for low-resource languages. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 583–592.

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.

Faheem Kirefu. 2018. *Exploring Context Representations for Neural Lemmatisation*. Master's thesis, University of Edinburgh.

Peter Makarov and Simon Clematide. 2018a. Imitation learning for neural morphological string transduction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2877–2882.

Peter Makarov and Simon Clematide. 2018b. Neural transition-based string transduction for limited-resource setting in morphology. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 83–93.

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.

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, Lisbon, Portugal. Association for Computational Linguistics.

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

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 157–163.

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.

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.

Richard Wicentowski. 2002. *Modeling and learning multilingual inflectional morphology in a minimally supervised framework*. Ph.D. thesis, Johns Hopkins University.

A Lematus Training

Lematus is implemented using the *Nematus* machine translation toolkit[10]. We use default training parameters of Lematus as specified by Bergmanis and Goldwater (2018) except for early stopping with patience (Prechelt, 1998) which we increase to 20. Similar to Bergmanis and Goldwater (2018) we use the first epochs as a burn-in period, after which we validate the current model by its lemmatization exact match accuracy on the *first 3k instances* of development set and save this model if it performs better than the previous best model. We choose a burn-in period of 20 and validation interval of 5 epochs for models that we train on datasets up to 2k instances and a burn-in period of 10 and validation interval of 2 epochs for others. As we work with considerably smaller datasets than Bergmanis and Goldwater (2018) we reduce the effective model size and increase the rate of convergence by tying the input embeddings of the encoder, the decoder and the softmax output embeddings (Press and Wolf, 2017).

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