Enhancing Sequence-to-Sequence Neural Lemmatization with External Resources

Kirill Milintsevich
Institute of Computer Science
University of Tartu
Tartu, Estonia
kirill.milintsevich@ut.ee

Kairit Sirts
Institute of Computer Science
University of Tartu
Tartu, Estonia
kairit.sirts@ut.ee

Abstract

We propose a novel hybrid approach to lemmatization[1] that enhances the seq2seq neural model with additional lemmas extracted from an external lexicon or a rule-based system. During training, the enhanced lemmatizer learns both to generate lemmas via a sequential decoder and copy the lemma characters from the external candidates supplied during run-time. Our lemmatizer enhanced with candidates extracted from the Apertium morphological analyzer achieves statistically significant improvements compared to baseline models not utilizing additional lemma information, achieves an average accuracy of 97.25% on a set of 23 UD languages, which is 0.55% higher than obtained with the Stanford Stanza model on the same set of languages. We also compare with other methods of integrating external data into lemmatization and show that our enhanced system performs considerably better than a simple lexicon extension method based on the Stanza system, and it achieves complementary improvements w.r.t. the data augmentation method.

1 Introduction

State-of-the-art lemmatization systems are based on attentional sequence-to-sequence neural architectures operating on characters that transform the surface word form into its lemma (Kanerva et al., 2018; Qi et al., 2018). Like any other supervised learning model, these systems are dependent on the amount and quality of the existing training data. Attempts to develop even more accurate lemmatization systems can focus on improving the model's architecture or obtaining additional data. While annotating additional data is an ongoing process for many smaller languages in the Universal Dependencies (UD) collection, there are also other data sources available that can be useful for improving lemmatization systems. In particular, we refer to existing rule-based morphological analyzers, lexicons, and other such resources.

Three potential sources for extracting additional lemma candidates are Apertium, Unimorph, and UD Lexicons initiatives. Apertium[2] is an open-source rule-based machine translation platform (Forcada et al., 2011). It also includes rule-based morphological analyzers based on finite-state transducers that cover 80 languages. Unimorph[3] is a project aimed at collecting annotated morphological inflection data, including lemmas, from Wiktionary (Kirov et al., 2016), a free open dictionary for many languages. Currently, the Unimorph project covers 110 languages. UD Lexicons[4] is a collection of 53 morphological lexicons in CoNLL-UL format covering 38 languages. UD Lexicons mostly use Apertium and Giellatekno systems to generate the annotations (Sagot, 2018).

Several previous works have proposed methods to improve lemmatization systems by augmenting the training data with additional instances (Bergmanis and Goldwater, 2019; Kanerva et al., 2020). In this paper, we propose another approach that both modifies the model architecture and leverages additional data. Unlike previous work where the model gains from extracting extra knowledge from the additional data provided for training, our primary goal is to teach the model to use external resources, even those that may only be available later during test time. In particular, the proposed system is a dual-encoder model, which receives two inputs for each word: 1) the word form itself to be lemmatized and 2) (optionally) the lemma candidates for that word form extracted from a lexicon or generated by a rule-based system. Both inputs are encoded

[1] https://github.com/501Good/lexicon-enhanced-lemmatization

[2] https://www.apertium.org

[3] http://unimorph.org/

[4] http://atoll.inria.fr/~sagot/

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pages 3112–3122
April 19 - 23, 2021.

with two different encoders and passed to the decoder. The decoder then learns via two separate attentional mechanisms to generate the lemma via the combination of the regular transduction and by copying characters from the external candidates. This way, the model is trained to use two sources of information–the regular training set and the options proposed by an external resource.

The experiments with several models enhanced with external data on 23 UD languages show that the best model using additional lemma candidates generated by the Apertium system achieves significantly higher results than the baseline models trained on the UD training set only. Also, we compare our method with other methods using external data. The enhanced system performs considerably better than a simple lexicon extension method based on the Stanza system, and it achieves complementary improvements w.r.t. the data augmentation method of Kanerva et al. (2020).

2 Related Works

Nowadays, state-of-the-art lemmatization systems are typically based on a neural sequence-to-sequence architecture, as demonstrated by the variety of systems presented at the CoNLL 2018 (Zeman et al., 2018) and SIGMORPHON 2019 (McCarthy et al., 2019) shared tasks. Several systems, including the TurkuNLP pipeline, the winner of the lemmatization track at CoNLL 2018 Shared task, use an attention-based translation model (Kanerva et al., 2018; Qi et al., 2018). The input to the system is the character sequence of a surface form (SF), which is "translated" into the lemma by an attention-based decoder. The input sequence can also be extended with POS tags (Qi et al., 2018) and morphological features (Kanerva et al., 2018).

Another approach was used by the UDPipe Future system, the second-best model at the CoNLL 2018 Shared Task. Straka (2018) proposed to produce a lemma by constructing a set of rules that transform the SF into a lemma. These rules can include copying, moving, or deleting a character in the SF, as well as additional rules for changing or preserving the casing. Thus, the lemmatization task is rendered into a multi-class classification task of choosing the correct transformation rule among the set of all possible rules generated from the training set. A year later, Straka et al. (2019) improved the result for the lemmatization by adding BERT contextual embeddings (Devlin et al., 2019) to the input, which made them the best lemmatization system at the SIGMORPHON 2019 Shared Task.

Several previous works have proposed to leverage additional data to improve lemmatization. In the simplest form, training data itself can be used to create a lexicon that maps word forms to its lemma. This strategy has been adopted by the Stanford neural lemmatization system (Qi et al., 2018), which creates such lexicons from the training sets and resorts to lemma generation only when the lexicon lookup fails. One can easily imagine extending such a lexicon with external resources. Rosa and Mareček (2018) adopted another simple way of using Unimorph lexicons to post-fix the morphological features and lemmas predicted by the UDPipe system (Straka and Straková, 2017). The post-fix is performed by simply looking up the SF from the Unimorph lexicon and, if the match is found, replacing the model prediction with the tags and lemmas found in the lexicon.

Another line of work has used additional data to augment the training data set. Bergmanis and Goldwater (2019) augmented their training set by first listing all non-ambiguous word-lemma pairs from Unimorph lexicons and then extracted sentences from Wikipedia that contained these words. They then trained the context-sensitive Lematus model (Bergmanis and Goldwater, 2018) on this extended partially lemmatized data set. Kanerva et al. (2018) used Apertium's morphological analyzer module to extend the training set for languages with tiny UD datasets. Apertium was used to generate all possible morphological analyses to 5000 sentences selected from the Wikipedia of the respective language. For each sentence, the most likely analysis sequence was then obtained via a disambiguating language model. The words that were assigned an Apertium-generated lemma during this process were added to the lemmatizer training set. In the subsequent work, Kanerva et al. (2020) extended the training data even more. They used Apertium to analyze all words found in the CoNLL 2017 web crawl dataset (Ginter et al., 2017) or in the Wikipedia of the respective language. All new words with unambiguous lemma and morphological analysis were added to the augmented training set.

3 Method

The core of the proposed model is the Stanford lemmatizer (Qi et al., 2018, 2020) which is a sequence-to-sequence model with attention. It takes character-

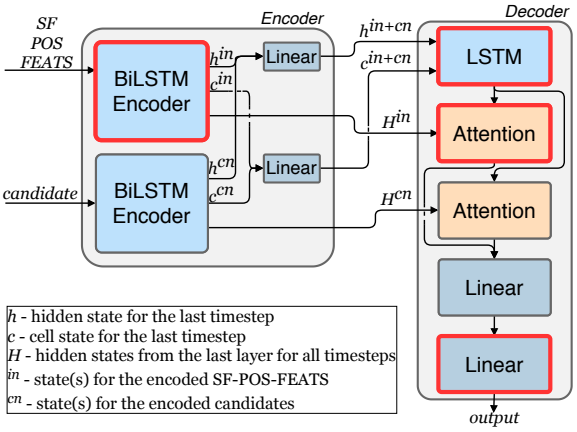

Figure 1: The architecture of the dual-encoder enhanced lemmatizer. Layers that comprise the original Stanza lemmatizer are marked with a bold red border.

level word representation and the POS tag as input and processes them with a bidirectional LSTM encoder. Then, it passes the encoder outputs to an LSTM decoder, which applies a soft dot attention layer after every LSTM cell. Finally, the output is constructed via greedy decoding.

We make several changes to the model architecture as shown in Figure 1. The components comprising the original Stanford lemmatizer are marked on the figure with the bold red border. First, we add another encoder that encodes the lemma candidates provided by the external system. The output representations of both encoders are combined with a linear layer and fed to the decoder. Secondly, we add another attention layer to the decoder that attends to the outputs of the second encoder. The outputs are finally combined with a linear layer. Finally, in addition to the POS tag, we also add morphological features to the first encoder's input.

Additionally, we implement the encoder dropout to simulate the situation when the external candidates are absent. The value of the encoder dropout that varies in the range of $\{0.0, 1.0\}$ defines the probability of discarding all candidates from a batch during training. Thus, the model will train only the main encoder based on this batch. This helps to train the model to perform more robustly in both situations when the candidates in the second encoder are present or absent.

4 Experiments

Data The models are trained and tested on the Universal Dependencies (UD) v2.5 corpora (Zeman et al., 2019). As additional external data, the lexicons from the Unimorph project (Kirov et al., 2016), UD Lexicons (Sagot, 2018), and lemmas generated with the Apertium morphological analyzer module (Forcada et al., 2011) are used. We also experiment with the lexicon constructed from the training set to simulate the situation when no additional data is available—this scenario assesses the effect of the second encoder without external data. The experiments are conducted on 23 languages from the UD collection. The basis of this selection was that all these languages are supported by both Unimorph, UD Lexicons, and Apertium.

To extract lemmas from the Unimorph lexicon, the input surface form (SF) is queried from the lexicon to retrieve the corresponding lemma. Some morphological forms in the Unimorph lexicons consist of several space-separated tokens; these were discarded. UD Lexicons are presented in the CoNLL-UL format, which is an extension of the CoNLL-U format. This makes the extraction process trivial since the lexicons are already pretokenized. For Apertium, all generated lemmas were stripped from special annotation symbols, and duplicate lemmas were removed. Finally, the simple training set based lexicon solution, similar to Qi et al. (2018), consists of two lookup dictionaries. The first lexicon maps SF-POS pairs to their lemmas, the second lexicon maps just SF's to their possible lemmas found in the training set. The lemma candidates for a SF are selected by first querying the input SF and POS tag from the SF-POS dictionary and, in case of failure, falling back to the SF dictionary.

Baselines As the first baseline, we compare our results with Stanza, the lemmatization module from the Stanza pipeline (Qi et al., 2020), which is a repackaging of the Stanford lemmatization system from the CoNLL 2018 Shared Task (Qi et al., 2018). We used the lemmatization models trained on the UDv2.5 available on the Stanza web page. As the Default baseline, we use our enhanced model, with the second encoder always being empty.

Experimental Setup We train four enhanced dual-encoder models that differ in the input to the second encoder. For all models, the input to the first encoder is the concatenation of SF characters, POS tag, and morphological features. During the training phase, gold POS tags and morphological features are supplied, while during inference, POS tags predicted with the Stanza tagger are used. The input to the second encoder is the following: for the second baseline (Default), it is always empty; for

Treebank	Size	All words					Out-of-vocabulary			
		Def	Lex	Uni	Apt	Stanza	Def	Apt	Diff	OOV%
cs_pdt	1,503K	98.51	98.66	**98.67**	98.55	98.58	90.51	90.95	0.44	7.53
ru_syntagrus	1,107K	97.82	97.92	98.00	**98.11**	97.91	89.48	91.65	2.17	10.56
es_ancora	547K	99.31	99.28	99.31	**99.35**	99.21	95.06	95.40	0.34	5.90
ca_ancora	530K	98.85	98.83	98.83	**98.89**	98.49	95.82	96.79	0.97	5.43
fr_gsd	389K	98.05	98.07	98.10	98.13	**98.15**	89.50	90.83	1.33	6.19
hi_hdtb	351K	98.77	98.71	98.71	**98.80**	96.66	93.49	94.62	1.13	4.67
de_gsd	287K	96.87	96.91	**97.04**	96.80	96.78	85.53	85.22	-0.31	13.04
it_isdt	278K	98.19	98.39	98.31	**98.48**	98.32	90.28	92.22	1.94	5.86
en_ewt	254K	98.21	98.19	98.22	**98.26**	98.18	90.10	90.49	0.39	10.05
ro_rrt	218K	98.33	98.28	98.32	**98.53**	98.16	91.46	93.22	1.76	11.60
pt_bosque	210K	98.24	98.20	98.23	**98.32**	98.12	93.15	94.27	1.12	8.85
nl_alpino	208K	**97.08**	96.61	96.89	96.74	96.99	86.34	84.88	-1.46	15.81
bg_btb	156K	97.97	**98.20**	98.17	98.07	97.36	91.07	91.02	0.05	13.97
ur_udtb	138K	97.16	**97.29**	97.28	97.28	95.62	91.83	91.93	0.01	6.79
gl_ctg	126K	98.48	98.48	98.51	**98.93**	98.59	89.73	93.55	3.82	10.94
uk_iu	122K	97.03	97.07	97.06	**97.12**	96.70	91.15	91.32	0.17	33.62
eu_bdt	121K	96.48	96.62	96.63	**96.68**	96.52	86.18	86.81	0.63	21.68
da_ddt	100K	97.87	97.7	97.81	**98.03**	97.36	89.86	90.31	0.45	18.13
sv_talbanken	96K	97.36	97.59	97.64	**98.27**	97.53	87.66	92.33	4.67	17.52
el_gdt	61K	96.84	97.06	97.25	**97.38**	96.66	84.18	86.42	2.24	19.59
tr_imst	56K	97.03	97.23	97.13	**97.39**	96.73	92.27	93.18	0.91	36.25
hy_armtdp	52K	95.55	95.84	94.87	**96.01**	95.55	86.11	87.34	1.23	38.54
be_hse	13K	81.91	81.86	82.36	**82.63**	79.98	68.78	70.30	1.52	93.28
Average		97.04	97.09	97.10	**97.25**	96.70	89.11	90.22	1.11	

Table 1: Lemmatization accuracy of the models enhanced with training the set lexicon (Lex), Unimorph lexicon (Uni), and Apertium systems (Apt) as well as the Default (Def)and Stanza baselines on 23 UD languages.

the Lexicon, Unimorph, and Apertium enhanced models, it contains the lemma candidate(s) from the training set based lexicon, Unimorph lexicons, and Apertium analyses respectively. If several possible candidates are returned for a SF, then these are concatenated. The encoder dropout for the Lexicon model is set to 0.8 to simulate the situation during testing for out-of-vocabulary (OOV) words where the second encoder will be empty. All models were trained in the HPC at the University of Tartu (University of Tartu, 2018) for a maximum of 60 epochs with stopping early if there was no improvement in the development accuracy in 10 epochs.

5 Results

Table 1 shows the results for all three enhanced systems and two baselines. The Apertium model outperforms other models for most languages, although the absolute differences are quite small. The Lexicon model and the Default baseline are on the same level on average, suggesting that supplying the model with lemmas extracted from the training set via the second encoder does not help to leverage the training data better. However, all enhanced models, including the Default model, perform better than the Stanza baseline, suggesting that omitting the lexicon heuristics and supplying the input tokens with both POS and morphological features might improve performance.

One-way ANOVA was performed to detect statistical difference between the systems.[5] A significant difference between the scores at the $p < 0.05$ level ($p = 0.038$) was found. Post hoc comparisons using one-sided paired t-tests showed that the mean accuracy of the Apertium-enhanced model is significantly greater compared to the the Default ($p_{adj} = 0.0005$), Lexicon

[5]The results for be_hse were extreme outliers and were not included in the comparison. The Unimorph-enhanced model was excluded from this test as its results did not conform to the normality requirement.

($p_{adj} < 0.0001$), UNIMORPH ($p_{adj} = 0.0001$) and STANZA ($p_{adj} < 0.0001$) systems with the p-value adjusted for multiple comparisons using the Bonferroni correction.

As the baseline model performances are already very high and the external information is expected to improve the lemmatization most for the new words unseen during training, we computed the accuracy of the out-of-vocabulary words (OOV) for the best performing APERTIUM model and the DEFAULT baseline. In this context, OOV words are those words in the test set that were not seen by the model during training. The results are shown in the right-most section of the Table 1. The improvements on the OOV words are variable, depending on the language, although on average, the improvement of the APERTIUM model over the DEFAULT baseline is more than 1%. We hypothesize that the direction and the magnitude of these effects are dependent on the coverage and the quality of the Apertium morphological analyzer.

6 Analysis of the Results

In this section, we analyze more thoroughly the potential of the proposed method. First, we compare our enhanced system with alternative methods for deploying external data, particularly with the data augmentation method proposed by Kanerva et al. (2020) and a lexicon extension method implemented based on the Stanza system (Qi et al., 2020). Secondly, we present more analyses to provide evidence towards the conclusion that the improvements presented for the enhanced model in the previous section can be attributed to our system's ability to make use of external resources supplied to the model via the second encoder.

6.1 Data Augmentation

We implemented the transducer augmentation method described by Kanerva et al. (2020). This method's basic idea relies on applying existing morphological analyzers (in this case, Apertium) to unannotated data to generate additional training instances. To obtain the augmentation data, we recreated the experiments of Kanerva et al. (2020) with 8K additional data. First, we collected a word frequency list for each language based on automatically annotated CoNLL2017 corpora (Ginter et al., 2017). For the languages not present in this dataset (Belarusian and Armenian), we used the wikidump to extract the word frequency list. Next, all words in the list were analyzed with the Apertium morphological analyzer. Then, we used the scripts[6] from the original experiments of Kanerva et al. (2020) to convert the Apertium analyses to the UD format and filter out ambiguous cases. Finally, the 8K most frequent words not already present in the training set together with their analyses were chosen and appended to the UD training set.

Although both the enhanced and augmented systems utilize Apertium as the external source, additional data usage differs. The augmented system uses Apertium to create extra labeled training data, while our enhanced model uses Apertium to generate additional lemma candidates to the words of the same initial training set. On the other hand, during test time, the augmented model must fully rely on the regularities learned during training, while our enhanced model can additionally look at the lemmas for words that were never seen during training.

The comparison of our APERTIUM-enhanced model and the augmented model is shown in the first two blocks of Table 2. The first two columns reintroduce the DEFAULT and APERTIUM-enhanced models' results from the Table 1, the third and the fourth columns show the same two models trained on the augmented training sets. Overall, the average results for both APERTIUM-enhanced and the augmented DEFAULT model (the column DEF+8K) are very similar, with the average of the APERTIUM-enhanced model being slightly higher (97.25 vs. 97.17). The APERTIUM-enhanced model is better in 15 languages out of 23 (underlined in the table), while the augmented model surpasses the enhanced model on 8 models. The APT+8K column shows the results of a model combining both augmentation and enhanced methods—the training data is first augmented with the additional 8K words and then additionally enhanced with the Apertium candidates via the second encoder. The combined approach scores are the best for 8 languages out of 23, resulting in an average improvement over the augmented DEFAULT model of 0.14% and over the APERTIUM-enhanced model of 0.06% in absolute. These results show that both augmentation and enhancement methods can contribute in complementary ways.

6.2 Lexicon Extension

Another simple baseline method for using external data is to use a lexicon or an external system first

[6] https://github.com/jmnybl/universal-lemmatizer

Treebank	Def	Apt	Def+8K	Apt+8K	$\text{Apt}_{0.8}$	Apt+E	Apt+Uni	Apt+UD
	Our models		Augmented models		The second encoder input varies			
cs_pdt	98.51	98.55	98.49	**98.57**	98.49	98.39	98.51	98.50
ru_syntagrus	97.82	**98.11**	97.86	98.06	97.98	97.83	97.98	97.97
es_ancora	99.31	99.35	99.53	**99.60**	99.33	99.29	99.33	99.33
ca_ancora	98.85	**98.89**	98.86	**98.89**	98.85	98.80	98.85	98.85
fr_gsd	98.05	98.13	98.98	**99.05**	97.98	97.79	97.97	97.98
hi_hdtb	98.77	98.80	**98.83**	98.78	98.84	98.66	98.83	98.84
de_gsd	**96.87**	96.80	96.79	96.67	96.83	96.49	96.83	96.84
it_isdt	98.19	98.48	98.98	**98.99**	98.36	98.3	98.36	98.37
en_ewt	98.21	**98.26**	97.24	98.12	98.21	98.17	98.22	98.20
ro_rrt	98.33	**98.53**	97.56	98.48	98.44	98.29	98.46	98.41
pt_bosque	98.24	**98.32**	98.13	98.29	98.30	98.32	98.30	98.31
nl_alpino	97.08	96.74	96.80	**96.82**	96.89	96.86	96.81	96.85
bg_btb	97.97	98.07	**98.84**	98.82	98.02	98.06	98.02	98.02
ur_udtb	97.16	97.28	96.90	**97.31**	97.13	97.13	97.13	97.13
gl_ctg	98.48	**98.93**	98.27	98.84	98.74	97.02	98.74	98.70
uk_iu	97.03	97.12	97.25	**97.35**	97.22	97.11	97.22	97.22†
eu_bdt	96.48	96.68	96.66	**96.71**	96.63	96.33	96.63	96.62
da_ddt	97.87	**98.03**	97.74	97.95	97.87	97.57	97.91	97.87
sv_talbanken	97.36	**98.27**	97.49	98.16	98.41	97.64	97.84	97.95
el_gdt	96.66	**97.38**	97.02	96.96	97.49	97.38	97.56	97.47
tr_imst	97.03	**97.39**	97.01	97.24	97.17	96.89	97.17	97.14
hy_armtdp	95.55	**96.01**	95.74	95.66	95.86	95.68	95.86	95.86†
be_hse	81.91	82.63	**83.33**	82.92	83.51	82.13	83.51	83.51†
Average	97.03	97.25	97.17	**97.31**	97.24	96.96	97.22	97.21

Table 2: Comparison of the enhanced models with the augmentation method: Def is the Default model, Apt is the Apertium-enhanced model, Def+8K and Apt+8K are the same Default and Apertium-enhanced models with augmented data. For the models marked with †, the UD Lexicon is absent and is replaced with Apertium candidates instead.

and only resort to neural generation when the surface form (SF) is not present in the lexicon. This is essentially how the Stanza lemmatizer works. Stanza constructs a lexicon based on the training set. During inference, the prediction goes through a cascade of three steps: 1) if the SF is present in the lexicon, then the lemma is immediately retrieved from the lexicon. 2) If the SF is novel and is missing from the lexicon, an edit operation is generated that decides whether the SF itself or its lowered form is the lemma, or whether neither is true. 3) Only in the last case the lemma is generated by the sequential decoder. For testing out the lexicon extension system, we used the pretrained Stanza models but extended the lexicon stored in the Stanza system with additional items. Note that Stanza lexicons can only store one lemma per SF-POS combination. Thus, if any of the external lexicons contain ambiguous lemmas, the firstly encountered lemma is chosen for each word.

We extended the Stanza lexicons with both the Apertium 8K datasets used for training the augmented models in section 6.1 and the UD lexicons (Sagot, 2018). The results of these evaluations are shown in Table 3. The set of languages in this table is slightly different than in Table 1, only including those languages for which the UD lexicons are existent. The left block shows the results with various Stanza models. The first column shows the baseline Stanza results (taken from Table 1), the second and the third columns present the Stanza model with its lexicon extended with the UD lexicons and the 8K words, respectively. The original UD lexicon for Russian contained many erroneous lemmas due to poor post-processing, which skewed the average accuracy. Thus, we did additional post-processing to put it in line with other languages.

The average scores of the Stanza systems ex-

Treebank	Stanza	Stanza+UD	Stanza+8K	Apt	Lex+UD	Lex+8K
cs_pdt	98.58	**98.76**	98.60	98.49	98.70	98.66
ru_syntagrus	97.91	96.76†	97.92	**97.98**	97.36†	97.97
es_ancora	99.21	99.25	99.15	**99.33**	99.27	99.28
ca_ancora	98.49	98.29	98.51	**98.85**	98.83	98.83
fr_gsd	98.15	97.69	**98.24**	97.98	97.09	96.75
hi_hdtb	96.66	96.75	96.66	**98.84**	98.76	98.71
de_gsd	**96.78**	97.53	96.86	96.83	97.01	96.94
it_isdt	98.32	**98.60**	98.46	98.36	98.38	98.39
en_ewt	98.18	**98.21**	98.17	**98.21**	**98.21**	98.19
ro_rrt	98.16	**98.44**	98.27	**98.44**	98.38	98.28
pt_bosque	98.12	**98.32**	98.12	98.30	97.98	98.20
nl_alpino	96.99	**97.22**	96.97	96.89	96.63	96.61
bg_btb	97.36	96.26	96.62	98.02	98.12	**98.20**
ur_udtb	95.62	95.66	95.64	97.13	97.28	**97.29**
gl_ctg	98.59	98.64	98.60	**98.74**	98.48	98.48
eu_bdt	96.52	96.51	96.41	96.63	**96.66**	96.62
da_ddt	97.36	**97.89**	97.55	97.87	97.82	97.70
sv_talbanken	97.53	**98.45**	97.63	98.41	97.78	97.59
el_gdt	96.66	96.49	96.89	97.49	97.52	**97.54**
tr_imst	96.73	96.90	96.83	97.17	97.17	**97.23**
Average	97.60	97.63	97.61	**98.00**	97.87	97.87

Table 3: Evaluation of the effect of the Stanza-based lexicon extension method; comparison with the Apertium-enhanced (Apt) and the Lexicon-enhanced systems (Lex+UD and Lex+8K).

tended with both UD and 8K lexicons remain roughly the same. However, when extending the Stanza with UD lexicons, most languages improve at least slightly, as shown with the underlined scores in the column Stanza+UD. Overall, on average, the simple lexicon extension method falls considerably behind our Apertium-enhanced model (97.63 vs. 98.00), the scores of which are again replicated in the first column of the right-most block.

However, the Apertium-enhanced model is not directly comparable to the Stanza models with extended lexicons because 1) the training data differs as the enhanced model has access to extra lemma candidates of the training set words during training and 2) the lexicons available during the test time are different. Thus, we also show in the last two columns of the right-hand block of Table 3 the results of two Lexicon-enhanced models (recall Section 4 and Table 1), similarly extended with the UD and 8K lexicons. The Lexicon-enhanced model has access to the same data as the Stanza model during both training (training set + the training set based lexicon) and testing.

While the Lexicon-enhanced model alone does not perform better than the Default baseline (see results in Table 1), adopting additional UD or 8K lexicons during test time increases the results to the same level with the Apertium-enhanced model. This shows that our proposed approach does not need additional resources during training—the model can be trained to use external sources based on the lexicon created from the training set. Then, the system's real benefits can be achieved when using extra resources later during test time. Without those resources, the model still performs on the same level as the non-enhanced baseline.

We hypothesize that our dual-encoder approach performs better than the Stanza with extended lexicon partly because of the differences in the usage of the external data. Since Stanza uses the lexicon resources as a first step in the cascade, it is prone to potential errors and noise in the lexicons. The dual-encoder model is safer against noise in this respect because the lemma candidates are not simply chosen as the prediction if present but are rather fed through the system that can decide how much to take or ignore from the given candidates. Also, because Stanza lexicons have the restriction

of only one lemma per word-POS pair, the system might solve some ambiguities erroneously. Our approach is also more flexible in this respect, as the second encoder can be given several candidates, and again, the system learns to decide itself from which candidate how much to take. On average, there are 0.71 lemma candidates per input word, and 1.09 lemma candidates per input word when excluding those words that do not have external lemma candidates.

6.3 Effect of the Second Encoder

Next, we performed a set of evaluations to argue for the effect of the second encoder in the enhanced model. We suggest that the improvements presented in Table 1 for the Apertium-enhanced model over the Default baseline are indeed due to the input provided via the second encoder. To demonstrate that, we evaluated the test set for each language again, on the same model that was trained with Apertium lemma candidates but leaving the second encoder empty for the test time. For that, we retrained the Apertium-enhanced models with the encoder dropout of 0.8. This means that during training, 80% of the time, the lemma candidates provided for the second encoder are dropped, and the model trains only the main encoder. The reasoning for using the dropout is similar to one provided for the Lexicon-enhanced model in Section 3—if the lemma candidates are always provided during training, the model learns to rely equally on both encoders. Due to that, if the second encoder remains empty during testing, the performance degrades considerably. If, on the other hand, the dropout is used, then the model learns to make predictions both when the candidates in the second encoder are present and also when they are absent. The results of these experiments are shown in the right-most block of Table 2.

We first show in Table 2 that the results of the Apertium-enhanced models trained with dropout are equivalent to the results obtained without dropout as evidenced by the column $\text{Apt}_{0.8}$. Next, when the second encoder is empty (column Apt+E), the test results are similar to the ones obtained with the Default model, providing evidence that the improvements are indeed due to the extra info supplied via the second encoder during test time. Additionally, we emulated the scenario when extra lexicon information becomes available after training the model. In this case, it is straightforward to integrate this information into the system without having to retrain the model. The last two columns in Table 2 show the following scenarios on this respect: 1) Unimorph lexicons in addition to Apertium (7th column Apt+Uni) and 2) UD lexicons (the last column Apt+UD) in addition to Apertium. The results in Table 2 show that, on average, extending the Apertium system with these particular lexicons do not add any benefit. The reasons for that can be twofold: 1) The UD lexicons are for most languages constructed based on the Apertium system and thus might not add any extra information; 2) The coverage of Unimorph lexicons in terms of lemmas is typically smaller than of Apertium systems.

Table 4 shows some examples when the Default model predicted incorrect lemma while the Apertium-enhanced model predicted the correct one. In some cases, Apertium provided the only and correct candidate for the Apertium-enhanced model, which was picked as a final prediction. In other cases, several candidates are provided to the second encoder, and the enhanced model chooses the correct one in most of the cases. This indicates that the second encoder effectively learns how to use the candidates to better control the lemma generation.

6.4 Effect of Morphological Features

All dual-encoder models were trained with both POS and morphological features in the input, while the Stanza baseline only uses POS-tag information. Thus, the effect of the morphological features is a potential confounding factor when comparing the performance of the enhanced models to the Stanza baseline. To evaluate the effect of the morphological features, we trained the Default and Apertium-enhanced models with only providing POS-tag information to the input.

Figure 2 shows the improvement in accuracy over the Default model trained with POS-tags only of 1) the Default model trained with both POS-tags and morphological features, 2) the Apertium-enhanced model trained with only POS-tags, and 3) the Apertium-enhanced model trained with both POS-tags and morphological features. It can be seen that for some of the languages, the most improvement comes from adding morphological features to the input, while for other languages adding the second encoder gives the main boost. However, for most languages, combining the second encoder and morphological features provides the largest effect, which seems to be more complex than a linear combination of the two. We suppose that, in this scenario, the attention mechanism

Input	Def	Apt	Candidate(s)
папері ⟨*paperi*⟩ чотирьох ⟨*čotyr'oh*⟩	*папер ⟨*paper*⟩ *четвери ⟨*četvery*⟩	**папір** ⟨*papir*⟩ **чотири** ⟨*čotyry*⟩	папір ⟨*papir*⟩ четверо, чотири ⟨*četvero, čotyry*⟩
Antworten besten	*Antworte bester	**Antworten** **gut**	antworten, antwort gut
раскладзе ⟨*raskladze*⟩ стаіць ⟨*staic'*⟩	*раскладз ⟨*raskladz*⟩ стаіць ⟨*staic'*⟩	**расклад** ⟨*rasklad*⟩ **стаяць** ⟨*stajac'*⟩	раскласці, расклад ⟨*rasklasci, rasklad*⟩ стаяць, стаіць ⟨*stajac', staic'*⟩

Table 4: Examples for Ukrainian, German, and Belarusian words corrected by the enhanced model. All predictions of the Default (Def) are incorrect, the ungrammatical ones are marked with *. The correct predictions of the Apertium-enhanced (Apt) models are in **bold**. The last column shows the external candidates.

works differently—it allegedly takes the morphological features into account when picking the correct lemma from the multiple candidates.

7 Conclusion

We proposed a method for enhancing neural lemmatization by integrating external input into the model via a second encoder and showed that the system incorporating Apertium morphological analyzer significantly improved the performance over the baselines. Both Bergmanis and Goldwater (2019) and Kanerva et al. (2020) used external resources to augment the training data, and thus, the improvement of their system is dependent on the amount and quality of the extended data supplied during training. On the other hand, our method trains the system to use the external information provided during run-time, thus making it independent of the particular external data available during training.

We experimentally showed that the enhancing method is both slightly better and complementary to the data augmentation method of Kanerva et al. (2020). We also compared our system with a simple lexicon extension method implemented based on the Stanza system. When trained and tested in a comparable setting, the proposed enhanced system achieves considerably higher results.

Although the model's computational complexity is increased by introducing the second encoder, it is counterbalanced by our model being more robust to noise and the ambiguities stemming from the external lexicons. Moreover, the main bottleneck in computation originates not from the neural network's increased size but can rather stem from the external system. For example, in our experiments,

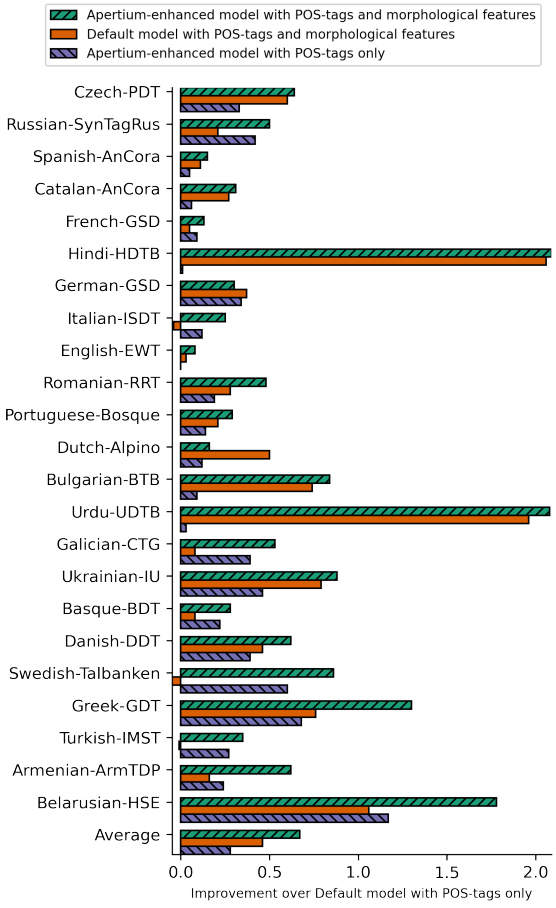

Figure 2: Independent and cumulative effects of the second encoder and the morphological features on the model's performance. The origin of the x-axis is the performance of the Default model with POS-tags only.

the main bottleneck in computation originated from executing the transducer-based Apertium morphological analyser. To overcome this bottleneck, one possible trade-off between the speed and accuracy is to precompile a candidate list large enough to cover the most frequent words for a given language. This is a problem that also simpler baseline methods adopting external resources have to address.

Finally, it is worth noting that the proposed method could be beneficial for less-resourced languages. However, establishing this claim would need more systematic experiments exploring specifically on this question, which we did not focus on in this paper. Still, because the significant improvements shown in this work are obtained on languages with larger datasets, the possible gains on smaller datasets can be larger.

Acknowledgments

The first author was supported by the IT Academy Program (StudyITin.ee).

References

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.

Toms Bergmanis and Sharon Goldwater. 2019. Training Data Augmentation for Context-Sensitive Neural Lemmatizer 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 1 (Long and Short Papers)*, pages 4119–4128.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Mikel L. Forcada, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M. Tyers. 2011. Apertium: A Free/Open-Source Platform for Rule-Based Machine Translation. *Machine translation*, 25(2):127–144.

Filip Ginter, Jan Hajic, Juhani Luotolahti, Milan Straka, and Daniel Zeman. 2017. CoNLL 2017 Shared Task–Automatically Annotated Raw Texts and Word Embeddings. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Jenna Kanerva, Filip Ginter, Niko Miekka, Akseli Leino, and Tapio Salakoski. 2018. Turku Neural Parser Pipeline: An End-to-End System for the CoNLL 2018 Shared Task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual parsing from raw text to universal dependencies*, pages 133–142.

Jenna Kanerva, Filip Ginter, and Tapio Salakoski. 2020. Universal Lemmatizer: A Sequence to Sequence Model for Lemmatizing Universal Dependencies Treebanks. *Natural Language Engineering*, pages 1–30.

Christo Kirov, John Sylak-Glassman, Roger Que, and David Yarowsky. 2016. Very-large Scale Parsing and Normalization of Wiktionary Morphological Paradigms. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Arya D. McCarthy, Ekaterina Vylomova, Shijie Wu, Chaitanya Malaviya, Lawrence Wolf-Sonkin, Garrett Nicolai, Christo Kirov, Miikka Silfverberg, Sebastian J. Mielke, Jeffrey Heinz, et al. 2019. The SIGMORPHON 2019 Shared Task: Morphological Analysis in Context and Cross-Lingual Transfer for Inflection. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–244.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D Manning. 2018. Universal Dependency Parsing from Scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Rudolf Rosa and David Mareček. 2018. CUNI x-ling: Parsing Under-Resourced Languages in CoNLL 2018 UD Shared Task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 187–196.

Benoît Sagot. 2018. A Multilingual Collection of CoNLL-U-Compatible Morphological Lexicons. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Milan Straka. 2018. UDPipe 2.0 Prototype at CoNLL 2018 UD Shared Task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207.

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, Vancouver, Canada. Association for Computational Linguistics.

Milan Straka, Jana Straková, and Jan Hajic. 2019. UDPipe at SIGMORPHON 2019: Contextualized Embeddings, Regularization with Morphological Categories, Corpora Merging. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 95–103.

University of Tartu. 2018. UT Rocket. share.neic.no.

Daniel Zeman, Jan Hajic, 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. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual parsing from raw text to universal dependencies*, pages 1–21.

Daniel Zeman, Joakim Nivre, Mitchell Abrams, and et al. 2019. Universal Dependencies 2.5. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.