A Simple Joint Model for Improved Contextual Neural Lemmatization

Abstract

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

Introduction

* Equal contribution. Listing order is random.

Lemmatization is a core NLP task that involves a string-to-string transduction from an inflected word form to its citation form, known as the lemma. More concretely, consider the English sentence: The bulls are running in Pamplona. A lemmatizer will seek to map each word to a form you may find in a dictionary—for instance, mapping running to run. This linguistic normalization is important in several downstream NLP applications, especially for highly inflected languages. Lemmatization has previously been shown to improve recall for information retrieval BIBREF0 , BIBREF1 , to aid machine translation BIBREF2 , BIBREF3 and is a core part of modern parsing systems BIBREF4 , BIBREF5 .

However, the task is quite nuanced as the proper choice of the lemma is context dependent. For

instance, in the sentence A running of the bulls took place in Pamplona, the word running is its own lemma, since, here, running is a noun rather than an inflected verb. Several counter-examples exist to this trend, as discussed in depth in haspelmath2013understanding. Thus, a good lemmatizer must make use of some representation of each word's sentential context. The research question in this work is, then, how do we design a lemmatization model that best extracts the morpho-syntax from the sentential context?

Recent work BIBREF7 has presented a system that directly summarizes the sentential context using a recurrent neural network to decide how to lemmatize. As N18-1126's system currently achieves state-of-the-art results, it must implicitly learn a contextual representation that encodes the necessary morpho-syntax, as such knowledge is requisite for the task. We contend, however, that rather than expecting the network to implicitly learn some notion of morpho-syntax, it is better to explicitly train a joint model to morphologically disambiguate and lemmatize. Indeed, to this end, we introduce a joint model for the introduction of morphology into a neural lemmatizer. A key feature of our model is its simplicity: Our contribution is to show how to stitch existing models together into a joint model, explaining how to train and decode the model. However, despite the model's simplicity, it still achieves a significant improvement over the state of the art on our target task: lemmatization.

Experimentally, our contributions are threefold. First, we show that our joint model achieves state-of-the-art results, outperforming (on average) all competing approaches on a 20-language subset of the Universal Dependencies (UD) corpora BIBREF8 . Second, by providing the joint model with gold morphological tags, we demonstrate that we are far from achieving the upper bound on performance—improvements on morphological tagging could lead to substantially better lemmatization. Finally, we provide a detailed error analysis indicating when and why morphological analysis helps lemmatization. We offer two tangible recommendations: one is better off using a joint model (i) for languages with fewer training data available and (ii) languages that have richer morphology.

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

Background: Lemmatization

Most languages BIBREF11 in the world exhibit a linguistic phenomenon known as inflectional morphology, which causes word forms to mutate according to the syntactic category of the word. The syntactic context in which the word form occurs determines which form is properly used. One privileged form in the set of inflections is called the lemma. We regard the lemma as a lexicographic convention, often used to better organize dictionaries. Thus, the choice of which inflected form is the lemma is motivated by tradition and convenience, e.g., the lemma is the infinitive for verbs in some Indo-European languages, Not in Latin. rather than by linguistic or cognitive concerns. Note that the stem differs from the lemma in that the stem may not be an actual inflection. In the NLP literature, the syntactic category that each inflected form encodes is called the morphological tag. The morphological tag generalizes traditional part-of-speech tags, enriching them with further linguistic knowledge such as tense, mood, and grammatical case. We call the individual key–attribute pairs morphological attributes.

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

A Joint Neural Model

The primary contribution of this paper is a joint model of morphological tagging and lemmatization. The intuition behind the joint model is simple: high-accuracy lemmatization requires a representation of the sentential context, in which the word occurs (this behind has been evinced in sec:introduction)—a

morphological tag provides the precise summary of the context required to choose the correct lemma. Armed with this, we define our joint model of lemmatization and morphological tagging as:

DISPLAYFORM0

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

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

We employ a simple LSTM-based tagger to recover the morphology of a sentence BIBREF12 , BIBREF13 . We also experimented with the neural conditional random field of P18-1247, but E17-1048 gave slightly better tagging scores on average and is faster to train. Given a sequence of INLINEFORM0 words INLINEFORM1 , we would like to obtain the morphological tags INLINEFORM2 for each word, where INLINEFORM3 . The model first obtains a word representation for each token using a character-level biLSTM BIBREF14 embedder, which is then input to a word-level biLSTM tagger that predicts tags for each word. Given a function cLSTM that returns the last hidden state of a character-based LSTM, first we obtain a word representation INLINEFORM4 for word INLINEFORM5 as,

DISPLAYFORM0

where INLINEFORM0 is the character sequence of the word. This representation INLINEFORM1 is then input to a word-level biLSTM tagger. The word-level biLSTM tagger predicts a tag from INLINEFORM2 . A full description of the model is found in

E17-1048[author=ryan,color=violet!40,size=,fancyline,caption=,]For camera ready, add citation to my paper. I removed it for anonymity.. We use standard cross-entropy loss for training this model and decode greedily while predicting the tags during test-time. Note that greedy decoding is optimal in this tagger as there is no interdependence between the tags INLINEFORM3 .

A Lemmatizer: p(ℓ i |m i ,w i)p(\ell _i \mid m_i, w_i)

Neural sequence-to-sequence models BIBREF15 , BIBREF16 have yielded state-of-the-art performance on the task of generating morphological variants—including the lemma—as evinced in several recent shared tasks on the subject BIBREF17 , BIBREF18 , BIBREF19 . Our lemmatization factor in eq:joint is based on such models. Specifically, we make use of a hard-attention mechanism BIBREF20 , BIBREF21 , rather than the original soft-attention mechanism. Our choice of hard attention is motivated by the performance of K18-3008's system at the CoNLL-SIGMORPHON task. We use a nearly identical model, but opt for an exact dynamic-programming-based inference scheme BIBREF22 .

We briefly describe the model here. Given an inflected word INLINEFORM0 and a tag INLINEFORM1 , we would like to obtain the lemma INLINEFORM2 , dropping the subscript for simplicity. Moreover, for the remainder of this section the subscripts will index into the character string INLINEFORM3 , that is INLINEFORM4 , where each INLINEFORM5 . A character-level biLSTM encoder embeds INLINEFORM6 to INLINEFORM7 . The decoder LSTM produces INLINEFORM8 , reading the concatenation of the embedding of the previous character INLINEFORM9 and the tag embedding INLINEFORM10 , which is produced by an order-invariant linear function. In contrast to soft attention, hard attention models the alignment distribution explicitly.

We denote INLINEFORM0 as the set of all monotonic alignments from INLINEFORM1 to INLINEFORM2 where an alignment aligns each target character INLINEFORM3 to exactly one source character in

INLINEFORM4 and for INLINEFORM5 , INLINEFORM6 refers to the event that the INLINEFORM7 character of INLINEFORM8 is aligned to the INLINEFORM9 character of INLINEFORM10 . We factor the probabilistic lemmatizer as, DISPLAYFORM0

The summation is computed with dynamic programming—specifically, using the forward algorithm for hidden Markov models BIBREF23 . INLINEFORM0 is a two-layer feed-forward network followed by a softmax. The transition INLINEFORM1 is the multiplicative attention function with INLINEFORM2 and INLINEFORM3 as input. To enforce monotonicity, INLINEFORM4 if INLINEFORM5 .

Decoding

We consider two manners, by which we decode our model. The first is a greedy decoding scheme. The second is a crunching BIBREF24 scheme. We describe each in turn.

In the greedy scheme, we select the best morphological tag sequence DISPLAYFORM0

and then decode each lemmata DISPLAYFORM0

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

In the crunching scheme, we first extract a INLINEFORM0 -best list of taggings from the morphological tagger. For an input sentence INLINEFORM1 , call the INLINEFORM2 -best tags for the INLINEFORM3 word INLINEFORM4 . Crunching then says we should decode in the following manner DISPLAYFORM0

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

Training with Jackknifing

In our model, a simple application of maximum-likelihood estimation (MLE) is unlikely to work well. The reason is that our model is a discriminative directed graphical model (as seen in fig:model) and, thus, suffers from exposure bias BIBREF25 . The intuition behind the poor performance of MLE is simple: the output of the lemmatizer depends on the output of the morphological tagger; as the lemmatizer has only ever seen correct morphological tags, it has never learned to adjust for the errors that will be made at the time of decoding. To compensate for this, we employ jackknifing BIBREF26 , which is standard practice in many NLP pipelines, such as dependency parsing.

Jackknifing for training NLP pipelines is quite similar to the oft-employed cross-validation. We divide our training data into INLINEFORM0 splits. Then, for each split INLINEFORM1 , we train the morphological tagger on the INLINEFORM2 split, and then decode it, using either greedy decoding or crunching, on the remaining INLINEFORM3 splits. This technique helps avoid exposure bias and improves the lemmatization performance, which we will demonstrate empirically in sec:exp. Indeed, the model is quite ineffective without this training regime. Note that we employ jackknifing for both the greedy decoding scheme and the crunching decoding scheme.

Dataset

To enable a fair comparison with N18-1126, we use the Universal Dependencies Treebanks BIBREF8 for all our experiments. Following previous work, we use v2.0 of the treebanks for all languages, except Dutch, for which v2.1 was used due to inconsistencies in v2.0. The standard splits are used for all

treebanks.

Training Setup and Hyperparameters

For the morphological tagger, we use the baseline implementation from P18-1247. This implementation uses an input layer and linear layer dimension of 128 and a 2-layer LSTM with a hidden layer dimension of 256. The Adam BIBREF27 optimizer is used for training and a dropout rate BIBREF28 of 0.3 is enforced during training. The tagger was trained for 10 epochs.

For the lemmatizer, we use a 2-layer biLSTM encoder and a 1-layer LSTM decoder with 400 hidden units. The dimensions of character and tag embedding are 200 and 40, respectively. We enforce a dropout rate of 0.4 in the embedding and encoder LSTM layers. The lemmatizer is also trained with Adam and the learning rate is 0.001. We halve the learning rate whenever the development log-likelihood increases and we perform early-stopping when the learning rate reaches INLINEFORM0 . We apply gradient clipping with a maximum gradient norm of 5.

Baselines (and Related Work)

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

The current state of the art is held by N18-1126, who, as discussed in sec:introduction, provide a direct context-to-lemma approach, avoiding the use of morphological tags. We remark that N18-1126 assume a setting where lemmata are annotated at the token level, but morphological tags are not available; we contend, however, that such a setting is not entirely realistic as almost all corpora annotated with lemmata at the token level include morpho-syntactic annotation, including the vast majority of the UD corpora. Thus, we do not consider it a stretch to assume the annotation of morphological tags to train our

joint model.

Our next baseline is the UDPipe system of K17-3009. Their system performs lemmatization using an averaged perceptron tagger that predicts a (lemma rule, UPOS) pair. Here, a lemma rule generates a lemma by removing parts of the word prefix/suffix and prepending and appending a new prefix/suffix. A guesser first produces correct lemma rules and the tagger is used to disambiguate from them.

The strongest non-neural baseline we consider is the system of D15-1272, who, like us, develop a joint model of morphological tagging lemmatization. In contrast to us, however, their model is globally normalized BIBREF29 . Due to their global normalization, they directly estimate the parameters of their model with MLE without worrying about exposure bias. However, in order to efficiently normalize the model, they heuristically limit the set of possible lemmata through the use of edit trees BIBREF30 , which makes the computation of the partition function tractable.

Much like D15-1272, Morfette relies on the concept of edit trees. However, a simple perceptron is used for classification with hand-crafted features. A full description of the model is given in grzegorz2008learning.

Results and Discussion

Experimentally, we aim to show three points. i) Our joint model (eq:joint) of morphological tagging and lemmatization achieves state-of-the-art accuracy; this builds on the findings of N18-1126, who show that context significantly helps neural lemmatization. Moreover, the upper bound for contextual lemmatizers that make use of morphological tags is much higher, indicating room for improved lemmatization with better morphological taggers. ii) We discuss a number of error patterns that the model seems to make on the languages, where absolute accuracy is lowest: Latvian, Estonian and Arabic. We suggest possible

paths forward to improve performance. iii) We offer an explanation for when our joint model does better than the context-to-lemma baseline. We show through a correlational study that our joint approach with morphological tagging helps the most in two cases: low-resource languages and morphologically rich languages.

Main Results

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

For measuring lemmatization performance, we measure the accuracy of guessing the lemmata correctly over an entire corpus. To demonstrate the effectiveness of our model in utilizing context and generalizing to unseen word forms, we follow N18-1126 and also report accuracies on tokens that are i) ambiguous, i.e., more than one lemmata exist for the same inflected form, ii) unseen, i.e., where the inflected form has not been seen in the training set, and iii) seen unambiguous, i.e., where the inflected form has only one lemma and is seen in the training set.

The results showing comparisons with all other methods are summarized in fig:results. Each bar represents the average accuracy across 20 languages. Our method achieves an average accuracy of INLINEFORM0 and the strongest baseline, N18-1126, achieves an average accuracy of INLINEFORM1 . The difference in performance (INLINEFORM2) is statistically significant with INLINEFORM3 under a paired permutation test. We outperform the strongest baseline in 11 out of 20 languages and underperform in only 3 languages with INLINEFORM4 . The difference between our method and all other baselines is statistical significant with INLINEFORM5 in all cases. We highlight two additional features of the data. First, decoding using gold morphological tags gives an accuracy of INLINEFORM6 for a

difference in performance of INLINEFORM7 . We take the large difference between the upper bound and the current performance of our model to indicate that improved morphological tagging is likely to significantly help lemmatization. Second, it is noteworthy that training with gold tags, but decoding with predicted tags, yields performance that is significantly worse than every baseline except for UDPipe. This speaks for the importance of jackknifing in the training of joint morphological tagger-lemmatizers that are directed and, therefore, suffer from exposure bias.

In fig:crunching, we observed crunching further improves performance of the greedy decoding scheme. In 8 out of 20 languages, the improvement is statistical significant with INLINEFORM0 . We select the best INLINEFORM1 for each language based on the development set.

In fig:error-analysis, we provide a language-wise breakdown of the performance of our model and the model of N18-1126. Our strongest improvements are seen in Latvian, Greek and Hungarian. When measuring performance solely over unseen inflected forms, we achieve even stronger gains over the baseline method in most languages. This demonstrates the generalization power of our model beyond word forms seen in the training set. In addition, our accuracies on ambiguous tokens are also seen to be higher than the baseline on average, with strong improvements on highly inflected languages such as Latvian and Russian. Finally, on seen unambiguous tokens, we note improvements that are similar across all languages.

Error Patterns

We attempt to identify systematic error patterns of our model in an effort to motivate future work. For this analysis, we compare predictions of our model and the gold lemmata on three languages with the weakest absolute performance: Estonian, Latvian and Arabic. First, we note the differences in the average lengths of gold lemmata in the tokens we guess incorrectly and all the tokens in the corpus. The

lemmata we guess incorrectly are on average 1.04 characters longer than the average length of all the lemmata in the corpus. We found that the length of the incorrect lemmata does not correlate strongly with their frequency. Next, we identify the most common set of edit operations in each language that would transform the incorrect hypothesis to the gold lemma. This set of edit operations was found to follow a power-law distribution.

For the case of Latvian, we find that the operation {replace: s INLINEFORM0 a} is the most common error made by our model. This operation corresponds to a possible issue in the Latvian treebank, where adjectives were marked with gendered lemmas. This issue has now been resolved in the latest version of the treebank. For Estonian, the operation {insert: m, insert: a} is the most common error. The suffix -ma in Estonian is used to indicate the infinitive form of verbs. Gold lemmata for verbs in Estonian are marked in their infinitive forms whereas our system predicts the stems of these verbs instead. These inflected forms are usually ambiguous and we believe that the model doesn't generalize well to different form-lemma pairs, partly due to fewer training data available for Estonian. This is an example of an error pattern that could be corrected using improved morphological information about the tokens. Finally, in Arabic, we find that the most common error pattern corresponds to a single ambiguous word form, 'an , which can be lemmatized as 'anna (like “that” in English) or 'an (like “to” in English) depending on the usage of the word in context. The word 'anna must be followed by a nominal sentence while 'an is followed by a verb. Hence, models that can incorporate rich contextual information would be able to avoid such errors.

Why our model performs better?

Simply presenting improved results does not entirely satiate our curiosity: we would also like to understand why our model performs better. Specifically, we have assumed an additional level of supervision—namely, the annotation of morphological tags. We provide the differences between our method and our retraining of the Lematus system presented in tab:diffs. In addition to the performance of

the systems, we also list the number of tokens in each treebank and the number of distinct morphological tags per language. We perform a correlational study, which is shown in tab:correlations.

We see that there is a moderate positive correlation (INLINEFORM0) between the number of morphological tags in a language and the improvement our model obtains. As we take the number of tags as a proxy for the morphological complexity in the language, we view this as an indication that attempting to directly extract the relevant morpho-syntactic information from the corpus is not as effective when there is more to learn. In such languages, we recommend exploiting the additional annotation to achieve better results.

The second correlation we find is a stronger negative correlation (INLINEFORM0) between the number of tokens available for training in the treebank and the gains in performance of our model over the baseline. This is further demonstrated by the learning curve plot in fig:learning, where we plot the validation accuracy on the Polish treebank for different sizes of the training set. The gap between the performance of our model and Lematus-ch20 is larger when fewer training data are available, especially for ambiguous tokens. This indicates that the incorporation of morphological tags into a model helps more in the low-resource setting. Indeed, this conclusion makes sense—neural networks are good at extracting features from text when there is a sufficiently large amount of data. However, in the low-resource case, we would expect direct supervision on the sort of features we desire to extract to work better. Thus, our second recommendation is to model tags jointly with lemmata when fewer training tokens are available. As we noted earlier, it is almost always the case that token-level annotation of lemmata comes with token-level annotation of morphological tags. In low-resource scenarios, a data augmentation approach such as the one proposed by BIBREF31 can be helpful and serve complementary to our approach.

Conclusion

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

Acknowledgments

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

Additional Results

We present the exact numbers on all languages to allow future papers to compare to our results in tab:dev and tab:test.