LemmaTag: Jointly Tagging and Lemmatizing for Morphologically Rich Languages with BRNNs

Daniel Kondratyuk[†] and **Tomáš Gavenčiak**[‡] and **Milan Straka**[†] and **Jan Hajič**[†]
Charles University
Faculty of Mathematics and Physics
[†]Institute of Formal and Applied Linguistics, [‡]Department of Applied Mathematics
dankondratyuk@gmail.com, gavento@kam.mff.cuni.cz, {straka,hajic}@ufal.mff.cuni.cz

Abstract

We present LemmaTag, a featureless neural network architecture that jointly generates part-of-speech tags and lemmas for sentences by using bidirectional RNNs with character-level and word-level embeddings. We demonstrate that both tasks benefit from sharing the encoding part of the network, predicting tag subcategories, and using the tagger output as an input to the lemmatizer. We evaluate our model across several languages with complex morphology, which surpasses state-of-the-art accuracy in both part-of-speech tagging and lemmatization in Czech, German, and Arabic.

1 Introduction

Morphologically rich languages are often difficult to process in many NLP tasks (Tsarfaty et al., 2010). As opposed to analytical languages like English, morphologically rich languages encode diverse sets of grammatical information within each word using inflections, which convey characteristics such as case, gender, and tense. The addition of several inflectional variants across many words dramatically increases the vocabulary size, which results in data sparsity and out-of-vocabulary (OOV) issues.

Due to these issues, morphological part-of-speech (POS) tagging and lemmatization are heavily used in NLP tasks such as machine translation (Fraser et al., 2012) and sentiment analysis (Abdul-Mageed et al., 2014). In morphologically rich languages, the POS tags typically consist of multiple morpho-syntactic subcategories providing additional information (see Figure 1). Closely related to POS tagging is lemmatization, which involves transforming each word to its root or dictionary form. Both tasks require context-sensitive awareness to disambiguate words with the same form but different syntactic or semantic features and behavior. Furthermore, lemmatization of a word form can benefit substantially from the information present in morphological tags, as grammatical attributes often disambiguate word forms using context (Müller et al., 2015).

We address context-sensitive POS tagging and lemmatization using a neural network model that jointly performs both tasks on each input word in a given sentence.[1] We train the model in a supervised fashion, requiring training data containing word forms, lemmas, and POS tags. In addition, we incorporate the ideas from Inoue et al. (2017) to optionally allow the network to predict the subcategories of each tag to improve accuracy. Our model is related to the work of Müller et al. (2015), which use conditional random fields (CRF) to jointly tag and lemmatize words for morphologically rich languages. The idea of jointly predicting several dimensions of categories has been explored prior to this work, for example, joint morphological and syntactic analysis (Bohnet et al., 2013) or joint parsing and semantic role labeling (Gesmundo et al., 2009).

Our model consists of three parts: (1) The **shared encoder**, which creates an internal representation for every word based on its character se-

[1]The code for this project is available at https://github.com/hyperparticle/LemmaTag

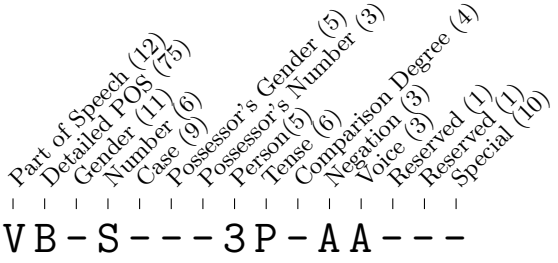


Figure 1: The tag components of the PDT Czech treebank with the numbers of valid values. Around 1500 different tags are in use in the PDT.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4921–4928
Brussels, Belgium, October 31 - November 4, 2018.

quence and the sentence context. We adopt the encoder architecture of Chakrabarty et al. (2017), utilizing character-level (Heigold et al., 2017) and word-level embeddings (Mikolov et al., 2013b; Santos and Zadrozny, 2014) processed through several layers of bidirectional recurrent neural networks (BRNN/BiRNN) (Schuster and Paliwal, 1997; Chakrabarty et al., 2017). (2) The **tagger decoder**, which applies a fully-connected layer to the outputs of the shared encoder to predict the POS tags. (3) The **lemmatizer decoder**, which applies an RNN sequence decoder to the combined outputs of the shared encoder and tagger decoder, producing a sequence of characters that predict each lemma (similar to Bergmanis and Goldwater (2018)).

The main advantages over other proposed models are: (i) The model is featureless, requiring little to no text preprocessing or morphological analysis postprocessing. (ii) The model shares the word embeddings, character embeddings, and RNN encoder weights in the tagger and lemmatizer, improving both tagging and lemmatization accuracy while reducing the number of parameters required for both tasks. (iii) The model predicts tag subcategories and provides the output of the tagger as features for the input of the lemmatizer, further improving accuracy.

We evaluate the accuracy of our model in POS tagging and lemmatization across several languages: Czech, Arabic, German, and English. For each language, we also compare the performance of a fully separate tagger and lemmatizer to the proposed joint model. Our results show that our joint model is able to improve the accuracy for both tasks, and achieves state-of-the-art performance in both POS tagging and lemmatization in Czech, German, and Arabic, while closely matching state-of-the-art performance for English.

2 The Joint LemmaTag Model

Given a sequence of words in a sentence $w_1, \ldots, w_k$, the task of the model is to produce a sequence of associated tags $t_1, \ldots, t_k$ and lemmas $\ell_1, \ldots, \ell_k$. For a word w_i at position i, we denote $c_{i,1}, c_{i,2} \ldots c_{i,m_i}$ to be the sequence of characters that make up w_i, where m_i indicates the length of the word string at position i. Analogously, we define $l_{i,1}, \ldots l_{i,\lambda_i}$ to be the sequence of characters that make up the lemma ℓ_i.

Our proposed model (shown in Figures 2 and 3) is split into three parts: the shared encoder, the tagger, and the lemmatizer. The initial layers of the model are shared between the tagger and lemmatizer, encoding the words, characters, and context in a given sentence. The encoder then passes its outputs to two networks, which perform a classification task to predict tags by the tagger and a sequence prediction task to output lemmas (character-by-character) in the lemmatizer.

2.1 Shared Encoder

In the encoder shown in Figure 2, each character $c_{i,1}, c_{i,2} \ldots c_{i,m_i}$ of a word w_i is indexed into an embedding layer to produce fixed-length embedded vectors representing each character. These vectors are further passed into a layer of BRNNs composed of gated recurrent units (GRU) (Cho et al., 2014) producing outputs $\mathbf{e}^c_1, \ldots, \mathbf{e}^c_m$, and whose final states are concatenated to produce the character-level embedding $\mathbf{s}^c_i$ of the word. Similarly, we index w_i into a word-level embedding layer to compute vector $\mathbf{e}^b_i$. Then we sum these results to produce the final word embedding $\mathbf{e}^w_i = \mathbf{s}^c_i + \mathbf{e}^b_i$.

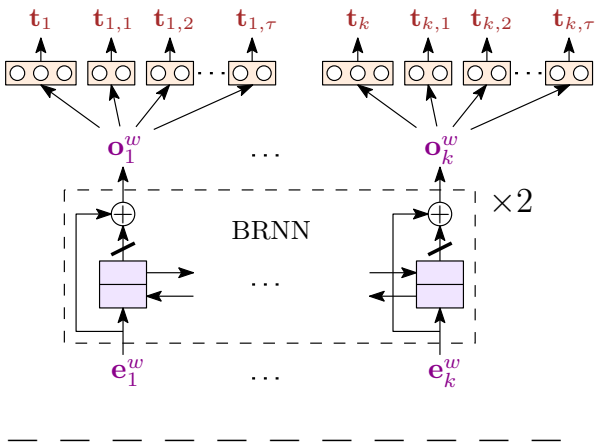
We repeat this process independently for all the words in the sentence and feed the resulting sequence $\mathbf{e}^w_1 \ldots \mathbf{e}^w_k$ into another two BRNN layers composed of long short-term memory units (LSTM) with residual connections. This produces word-level outputs $\mathbf{o}^w_1, \ldots \mathbf{o}^w_k$ that encode sentence-level context for each word (we ignore the final hidden states).

2.2 Tagger

The task of the tagger is to predict a tag $t_i \in \mathcal{T}$ given a word w_i and its context, where $\mathcal{T}$ is a set of possible tags. As explained the introduction, morphologically rich languages typically subdivide tags further into several subcategories $t_i = (t_{i,1}, \ldots, t_{i,\tau})$, where $t_{i,j} \in \mathcal{T}_j$, the j-th subcategory. See Figure 1 for an illustration taken from the Czech PDT tagset where $\tau = 15$.

Having the encoded words of a sentence available, the tagger consists of a fully-connected layer with $|\mathcal{T}|$ neurons whose input is the output of the word feature RNN $\mathbf{o}^w_i$ (see figure 2). This layer produces the logits $\mathbf{t}_i$ of the tag values and the predictions t_i as the maximum-likelihood value (i.e., softmax).

To obtain the information about categorical nature of each tag, we also predict every category $t_{i,j}$ of the tag independently (if they exist in the



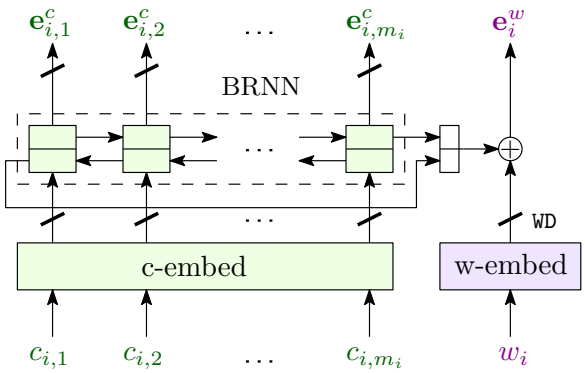


Figure 2: *Bottom:* Word-level encoder. The characters of every input word are embedded with a look-up table (c-embed) and encoded with a BRNN. The outputs $\mathbf{e}^c_{i,j}$ are used in decoder attention, and the final states are summed with the word-level embedding (w-embed) to produce $\mathbf{e}^w_i$. `WD` denotes word dropout.
Top: Sentence-level encoder and tag classifier. Two BRNN layers with residual connections act on the embedded words $\mathbf{e}^w_i$ of a sentence, providing context. The output of the tag classification are the logits for both the whole tags $\mathbf{t}_i$ and their components $\mathbf{t}_{i,j}$.
Both: Thick slanted lines denote training dropout.

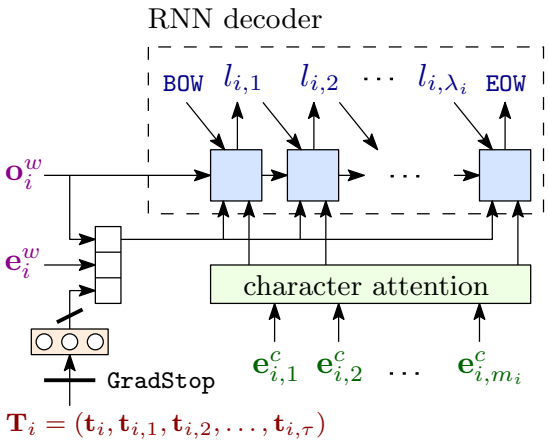


Figure 3: Lemma decoder, consisting of a standard seq2seq autoregressive decoder with Luong attention on character encodings, and with additional inputs of processed tagger features $\mathbf{T}_i$, embeddings $\mathbf{e}^w_i$ and sentence-level outputs $\mathbf{o}^w_i$. Gradients from the lemmatizer are stopped from flowing into the tagger (denoted `GradStop`).

dataset) with τ dense layers similar to Inoue et al. (2017). The j-th layer has $|\mathcal{T}_j|$ neurons and outputs the logits $\mathbf{t}_{i,j}$ for the category values. While these values are trained for, their value is not used in tag prediction. All tag values $\mathbf{T}_i = (\mathbf{t}_i, \mathbf{t}_{i,1} \ldots, \mathbf{t}_{i,\tau})$ are concatenated into a flat vector and fed into the lemmatizer as an additional set of potentially useful features.

2.3 Lemmatizer

The task of the lemmatizer is to produce a sequence of characters $l_{i,1}, \ldots, l_{i,\lambda_i}$ and the lemma length λ_i for each lemma ℓ_i. We use a recurrent sequence decoder, a setup typical of many sequence-to-sequence (seq2seq) tasks such as in neural machine translation (Sutskever et al., 2014).

The lemmatizer consists of a recurrent LSTM layer whose initial state is taken from word-level output $\mathbf{o}^w_i$ and whose inputs consist of three parts. The first part is the embedding of the previous output character (initially a *beginning-of-word* character `BOW`).

The second part is a character-level attention mechanism (Bahdanau et al., 2014) on the outputs of the character-level BRNN $\mathbf{e}^c_{i,1}, \ldots, \mathbf{e}^c_{i,m_i}$. We employ the multiplicative attention mechanism described in Luong et al. (2015), which allows the LSTM cell to compute an attention vector that selectively weights character-level information in $\mathbf{e}^c_{i,j}$ at each time step j based on the input state of the LSTM cell.

The third and final part of the RNN input allows the network to receive the information about the embedding of the word, the surrounding context of the sentence, and the output of the tagger. This output is the same for all time steps of a lemma and is a concatenation of the following: the output of the encoder $\mathbf{o}^w_i$, the embedded word $\mathbf{e}^w_i$ and processed tag features $\mathbf{T}^f_i$. The tag features are obtained by projecting the concatenated outputs of the tagger $\mathbf{T}_i$ through a fully connected layer with ReLU activation. During training, we do not pass the gradients back through $\mathbf{T}_i$ to prevent the distortion of the tagger output.

The decoder performs greedy decoding to predict the character outputs. It runs until it produces

the *end-of-word* character `EOW` or reaches a character limit of $m_i + 10$.

2.4 Loss Function

We define the final loss function as the weighted sum of the losses of the tagger and the lemmatizer:

$$L(\hat{y}, y) = \boldsymbol{\alpha} L(\hat{\mathbf{y}}^t, \mathbf{y}^t) + \beta L(\hat{\mathbf{y}}^\ell, \mathbf{y}^\ell)$$

where $\mathbf{y}$ are the predicted outputs, $\hat{\mathbf{y}}$ the expected outputs, $\mathbf{y}^t$, the tag components and $\mathbf{y}^\ell$ are the lemma characters. The tagger and lemmatizer losses are separately computed as the softmax cross entropy of the output logits. The weight hyperparameters $\boldsymbol{\alpha}, \beta$ scale the training losses so that the subtag and lemmatizer losses do not overpower the unfactored tag predictor gradients. The vector $\boldsymbol{\alpha}$ contains $\tau + 1$ weights: one for the whole tag and one for every component.[2]

3 Experiments

In this section, we show the outcomes of evaluation when running our joint tagger and lemmatizer and compare with the current state of the art in Czech, German, Arabic, and English datasets. Additionally, we evaluate the lemmatizer and tagger separately to compare the relative increase in tagging and lemmatization accuracy.

3.1 Datasets

Our datasets consist of the Czech Prague Dependency Treebank (PDT) (Hajič et al., 2006, 2018), the German TIGER corpus (Brants et al., 2004), the Universal Dependencies Prague Arabic Dependency Treebank (UD-PADT) (Hajic et al., 2004), the Universal Dependencies English Web Treebank (UD-EWT) (Silveira et al., 2014), and the WSJ portion of the English Penn Treebank (tags only) (Marcus et al., 1993). In all datasets, we use the tags specific to their respective language. Of these datasets, only Czech and Arabic provide subcategorical tags, and we use unfactored tags for the rest. See Table 1 for tagger and lemmatizer accuracies.

Note that the PDT dataset disambiguates lemmas with the same textual representation by appending a number as lemma sense indicator. For example, the dataset contains disambiguated lemmas `moc-1` (as *power*) and `moc-2` (as *too much*). About 17.5% of the PDT tokens have such sense-disambiguated lemmas. LemmaTag predicts the lemmas including the senses and the accuracies in Table 1 take that into account. Ignoring the sense ambiguity, the lemmatization accuracy of the joint LemmaTag model is 98.94% for Czech-PDT.

[2] If no components are available, $\tau = 0$.

3.2 Hyperparameters

We use loss weights $\boldsymbol{\alpha}_0 = 1.0$ for the whole tags, $\boldsymbol{\alpha}_{1,\ldots,\tau} = 0.1$ for the tag component losses and $\beta = 0.5$ for the lemmatizer loss.[3] The RNNs and word embedding tables have dimensionality 768 except for character-level embeddings and the character-level RNN, which are of dimension 384. The fully-connected layer whose inputs are $\mathbf{T}_i$ is of dimension 256.

We train the models for 40 epochs with random permutations of training sentences and batches of 16 sentences. The starting learning rate is $\eta = 0.001$ and we scale this by 0.25 at epochs 20 and 30 to increase accuracy. We train the network using the lazy variant of the Adam optimizer (Kingma and Ba, 2014), which only updates accumulators for variables that appear in the current batch (TensorFlow, 2018), with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$. We clip the global gradient norm to 3.0 to reduce the risk of exploding gradients.

To prevent the tagger from overfitting, we devise several strategies for regularization. We apply dropouts with rate 0.5 as indicated in Figures 2 and 3. The word dropout (`WD`) replaces 25% of words by the unknown token `<unk>` to force the network to rely more on context, combatting data sparsity issues. Lastly, we employ label smoothing (Pereyra et al., 2017) which is a way to prevent the network from being too confident in any one class. The label smoothing parameter is set to 0.1 for the tagger logits (both whole tags and the tag components).

Note that we did not perform any complex hyperparameter search. For additional information on real-world performance and additional techniques which have not improved evaluation accuracy, see Appendix A.

4 Conclusion

The evaluation results show that performing lemmatization and tagging jointly by sharing encoder parameters and utilizing tag features is

[3] These are reasonable values to prevent gradients from overpowering one another. The lemmatizer tends to influence the tagger heavily.

Approach	Czech-PDT*		German-TIGER		Arabic-PADT*		Eng-EWT		Eng-WSJ
	tag	lem	tag	lem	tag	lem	tag	lem	tag
LemmaTag (sep)	96.83	98.02	98.96	98.84	95.03	96.07	**95.50**	97.03	97.59
LemmaTag (joint)	**96.90**	**98.37**	**98.97**	**99.05**	**95.21**	**96.08**	95.37	**97.53**	N/A
SoTA results	95.89^{a}_{+}	97.86^{b}_{+}	98.04^{c}_{+}	98.24^{c}_{+}	91.68^{d}_{+}	92.60^{e}	93.90^{e}	96.90^{e}	$\mathbf{97.78}^{f}_{+}$

Table 1: Final accuracies on the test sets comparing the LemmaTag architecture as described (joint), LemmaTag neither sharing the encoder nor providing tagger features (sep), and the state-of-the-art results (SoTA). The state-of-the-art results are taken from the following papers: *(a)* Hajič et al. (2009), *(b)* Straková et al. (2014), *(c)* Eger et al. (2016), *(d)* Inoue et al. (2017), *(e)* Straka et al. (2016), *(f)* Ling et al. (2015). The results marked with a plus$_+$ use additional resources apart from the dataset, and datasets marked with a star* indicate the availability of subcategorical tags.

mutually beneficial in morphologically rich languages. We have shown that incorporating these ideas results in excellent performance, surpassing state-of-the-art in Czech, German, and Arabic POS tagging and lemmatization by a substantial margin, while closely matching state-of-the-art English POS tagging accuracy.

However, in languages with weak morphology such as English (and German to a lesser extent), sharing the encoder parameters may even hurt the performance of the tagger. We believe this is a consequence of tags correlating less with word-level morphology, and more with sentence-level syntax in morphologically poor languages. Lemma prediction could benefit from the syntactic information in the tags, but the tag predictions rely more on syntactic structure (i.e., word order) rather than on root forms of individual words which could be ambiguous.

There are some possible performance improvements and additional metrics which we leave for future work. For simplicity, one improvement we intentionally left out is the use of additional data. We can incorporate word2vec (Mikolov et al., 2013a) or ELMo (Peters et al., 2018) word representations, which have shown to reduce out-of-domain issues and provide semantic information (Eger et al., 2016). A second improvement is to integrate information from a morphological dictionary to resolve certain ambiguities (Hajič et al., 2009; Inoue et al., 2017). A third improvement can be to replace the seq2seq lemmatizer decoder with a classifier that chooses a corresponding edit tree to modify (reduce) the word form to its lemma (Chakrabarty et al., 2017). A fourth possible improvement would be to experiment with the Transformer model (Vaswani et al., 2017), which utilizes non-recurrent multi-headed self-attention and has been shown to achieve state-of-the-art performance in several related sequence tasks (Dehghani et al., 2018). Lastly, we would like to evaluate LemmaTag on a wider range of languages, e.g., on the Universal Dependencies (Nivre et al., 2016) languages and treebanks which employ lemmatization, and to analyze the use of different types of POS tags in the model.

The code we used for LemmaTag is available at `https://github.com/hyperparticle/LemmaTag`.

Acknowledgments

The work described herein has been supported by the City of Prague under the "OP PPR" program, project No. CZ.07.1.02/0.0/0.0/16_023/0000108 and it has been using language resources developed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071).

Tomáš Gavenčiak has been supported by Czech Science Foundation (GACR) project 17-10090Y "Network optimization". Daniel Kondratyuk has been supported by the Erasmus Mundus program in Language & Communication Technologies (LCT).

References

Muhammad Abdul-Mageed, Mona Diab, and Sandra Kübler. 2014. Samar: Subjectivity and sentiment analysis for arabic social media. *Computer Speech & Language*, 28(1):20–37.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

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), volume 1, pages 1391–1400.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richrd Farkas, Filip Ginter, and Ja n Haji. 2013. Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics*, 1:415–428.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. Tiger: Linguistic interpretation of a german corpus. *Research on language and computation*, 2(4):597–620.

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)*, volume 1, pages 1481–1491.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.

Steffen Eger, Rüdiger Gleim, and Alexander Mehler. 2016. Lemmatization and morphological tagging in german and latin: A comparison and a survey of the state-of-the-art. In *LREC*.

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

Andrea Gesmundo, James Henderson, Paola Merlo, and Ivan Titov. 2009. A latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, pages 37–42, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jan Hajič, Eduard Bejček, Alevtina Bémová, Eva Buráňová, Eva Hajičová, Jiří Havelka, Petr Homola, Jiří Kárník, Václava Kettnerová, Natalia Klyueva, Veronika Kolářová, Lucie Kučová, Markéta Lopatková, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Petr Pajas, Jarmila Panevová, Lucie Poláková, Magdaléna Rysová, Petr Sgall, Johanka Spoustová, Pavel Straňák, Pavlína Synková, Magda Ševčíková, Jan Štěpánek, Zdeňka Urešová, Barbora Vidová Hladká, Daniel Zeman, Šárka Zikánová, and Zdeněk Žabokrtský. 2018. Prague dependency treebank 3.5. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL, http://hdl.handle.net/11234/1-2621), Faculty of Mathematics and Physics, Charles University.

Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský, Magda Ševčíková-Razímová, and Zdeňka Urešová. 2006. Prague Dependency Treebank 2.0 (PDT 2.0). LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Jan Hajič, Jan Raab, Miroslav Spousta, et al. 2009. Semi-supervised training for the averaged perceptron pos tagger. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 763–771. Association for Computational Linguistics.

Jan Hajic, Otakar Smrz, Petr Zemánek, Jan Šnaidauf, and Emanuel Beška. 2004. Prague arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

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

Go Inoue, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Joint prediction of morphosyntactic categories for fine-grained arabic part-of-speech tagging exploiting tag dictionary information. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 421–431.

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

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Milan Straka, Jan Hajic, and Jana Strakov. 2016. Udpipe: Trainable pipeline for processing conll-u files performing tokenization, morphological analysis, pos tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Jana Straková, Milan Straka, and Jan Hajič. 2014. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

TensorFlow. 2018. tf.contrib.opt.lazyadamoptimizer: Class lazyadamoptimizer. TensorFlow documentation from tensorflow.org.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kübler, Marie Candito, Jennifer Foster, Yannick Versley, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing of morphologically rich languages (spmrl): what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

A Appendix

A.1 GPU Performance

We ran all the tests on an NVIDIA GTX 1080 Ti GPU. The joint LemmaTag training takes about 3 hours for Arabic PADT, 4.5 hours for English EWT, 12 hours for German TIGER, and 22 hours for Czech PDT. The separate models take about 50% more time. After training, the lemma and tag predictions of 219,000 test tokens of the Czech PDT take about 100 seconds.

A.2 Other Techniques

We briefly summarize some of the additional techniques we have tried but which do not improve the results. While some of those techniques do help on smaller models or earlier in the training, the effect on the fully trained network seems to be marginal or even detrimental.

Separate sense prediction. Instead of predicting the sense disambiguation with the lemmatizer (Czech only), we tried to predict sense as an additional classification problem with one dense layer based on $\mathbf{o}_i^w$ and $\mathbf{T}_i$, but it seems to perform slightly worse (0.2%).

Beam search decoder. We have implemented a beam search decoder for the lemmatizer instead of

the standard greedy one, but the improvement was marginal (around 0.01%).

Variational dropout. While the dropouts in the LemmaTag are completely random, variational dropout erases the same channels across the time steps of the RNN. While this generally improves training in convolutional networks and RNNs, we saw no significant difference.

Layer normalization. Layer normalization applied to the encoding RNNs did not bring significant gain and also slowed down the training.