Scaling Character-Based Morphological Tagging to Fourteen Languages

Georg Heigold and Josef van Genabith
DFKI & Saarland University
Saarbrücken, Germany
heigold@dfki.de, josef.van_genabith@dfki.de

Günter Neumann
DFKI
Saarbrücken, Germany
neumann@dfki.de

***Abstract*—This paper investigates neural character-based morphological tagging for languages with complex morphology and large tag sets. Character-based approaches are attractive as they can handle rarely- and unseen words gracefully. More specifically, beside a rich morphology, non-canonical language, change of language or other linguistic variability can heavily degrade the accuracy of natural language processing of web and CMC data. We evaluate on 14 languages and observe consistent gains over a state-of-the-art morphological tagger across all languages except for English and French, where we match the state-of-the-art. The gains are clearly correlated with the amount of training data. We present supplementary experiments to explore whether and to what extent unsupervised data through pre-trained word vectors can compensate for limited amounts of supervised data. Moreover, we show preliminary results to study the effect of noisy input data by flipping characters at random.**

***Keywords*-neural network architectures; recurrent neural networks; natural languages; morphological tagging**

I. Introduction

Character-based approaches have been studied for many applications in natural language processing, including part-of-speech (POS) tagging [1], [2], [3], [4], [5], morphological tagging [6], parsing [7], named entity recognition [3], language modeling [2], [8], and neural machine translation [9]. Character-based representations have the advantage of gracefully handling rare or unseen words and tend to produce more compact models as the number of atomic units, i.e., characters, is smaller compared to the number of words in word-level approaches. The issue of rare or unseen words is particularly pronounced when working on morphologically-rich languages, small amounts of training data or noisy user input.

Morphological tagging is the task of assigning a morphological analysis to a token in context. The morphological analysis for a word consists of a sequence of feature:value pairs describing, for example, case, gender, person and tense. A particular concatenation of such feature:value pairs is referred to as a single tag [10], [11], [12].

Following [13], we also add the part-of-speech to this morphological tag and refer to it as POS-MORPH:

```
I  see  four        words
                      |
```
POS=noun:CASE=acc:$\cdots$
$\cdots$:NUMBER=plural

Given a word in context, we predict a POS-MORPH tag as a complete unit, rather than as the individual component parts. This approach allows us to share large parts of the model but can only produce POS-MORPH analyses attested in the training data (cf. Table II). This is still the standard approach to morphological tagging and disambiguation as, given sufficient amounts of training data, the number of POS-MORPH descriptions that cannot be produced usually is small.

Character-based POS tagging (rather than full POS-MORPH tagging) has been extensively evaluated in the literature [1], [2], [3], [4]. The results are competitive but do not systematically outperform the state of the art. Only [4] report consistent gains by using shallow neural network architectures in combination with multitask learning, multilingual learning, and pre-trained word embeddings.

State-of-the-art results for morphological tagging (full POS-MORPH tagging) can be found in [12], [13]. To the best of our knowledge, there has not been much research on character-based morphological tagging so far. [6] is an exception but report results for German only. Their best results are on a par with state-of-the-art results. [14] show clear gains of character-based over state-of-the-art morphological taggers. However, the evaluation is limited to German and Czech.

Research on character-based approaches in general NLP clearly divides into papers that use CNN-based architectures [1], [8], [9] and papers that use LSTM-based architectures [6], [2], [3], [7], [4], [5]. There are a number of examples where an LSTM paper reports results of a CNN paper for comparison, such as [2] (POS tagging for English) and [3] (named entity recognition for English). However, there is no direct comparison between CNN and LSTM based architectures in morphological tagging.

Finally, several authors have shown the utility of word embeddings pre-trained on large amounts of unsupervised data [13], [2], [4], [5]. However, they do not discuss how the results obtained are impacted by the amounts of supervised

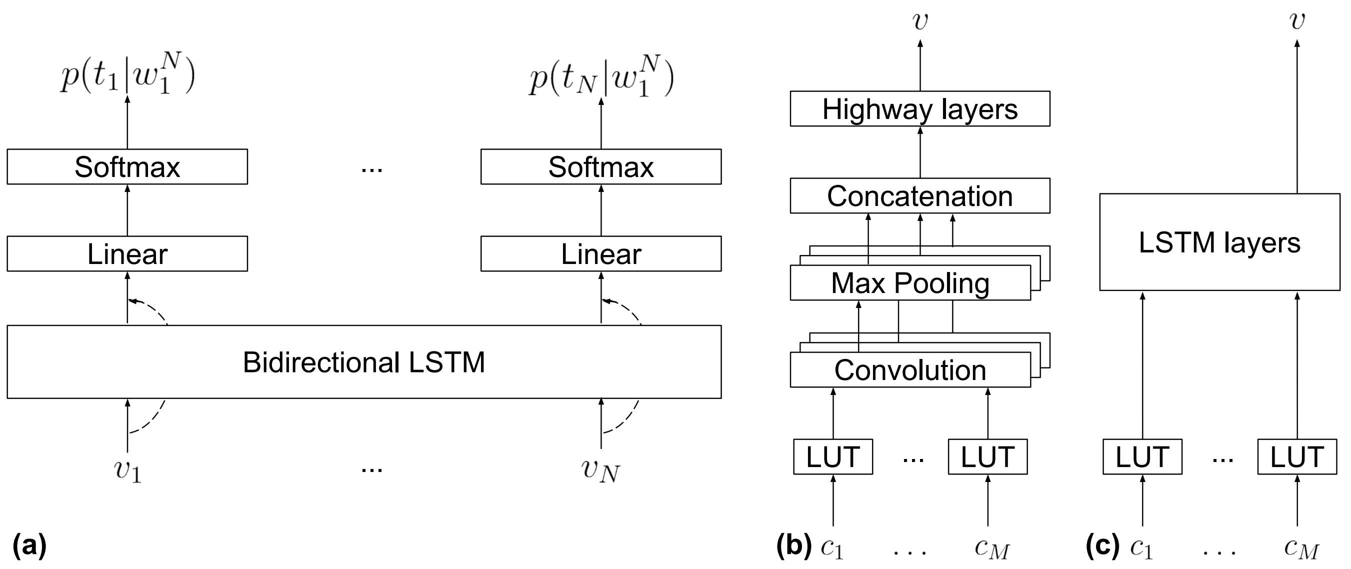

Figure 1. Character-based neural tagging architecture: (a) sub-network mapping word vectors v_1^N to tags t_1^N, dashed arrows indicate optional skip connections, (b) CNNs with different filter widths followed by fully-connected layers with highway connections (CNNHighway), and (c) deep LSTM using the last output to map the character string to a word vector. The networks in (b) and (c) are cloned to produce the input word vectors v_1^N in (a). LUT stands for lookup table.

data available: more precisely, do word vectors obtained from unlabeled supplementary data help in data settings where large amounts of labeled data are available?

In this paper, we investigate character-based morphological tagging in more depth. More specifically, the contributions of this paper include:

- the evaluation of character-based morphological tagging on 14 different languages of different morphological complexity;
- the demonstration of systematic gains of our character-based, language-agnostic morphological tagger over a state-of-the-art morphological tagger across morphologically rich languages. Moreover, and perhaps as expected, we show that the relative gains are clearly correlated with the amount of the training data;
- a preliminary study on the effect of artificial noisy input data;
- the evaluation of large amounts of unsupervised supplementary data through pre-trained word vectors to further explore the data-efficiency of the character-based approach.

The remainder of the paper is organized as follows. Section II summarizes the character-based neural network approaches used in this paper. The data sets and model configurations are described in Section III and in Section IV, respectively. The empirical evaluation is presented in Section V. Section VI concludes the paper. The appendix contains a listing of all experimental results obtained in this paper.

II. Character-based Tagging

We assume an input sentence w_1^N with (complex POS-MORPH morphological) output tags t_1^N and a zeroth-order Markov model

$$p(t_1^N|w_1^N) \quad = \quad \prod_{n=1}^{N} p(t_n|w_1^N) \qquad (1)$$

whose factors are modeled by a suitable neural network. For character-based tagging, we use the character representation of the word, $w = c_1^M$. This assumes that the segmentation of the sentence into words is known, which is straightforward for the languages under consideration.

At the top level, each input word maps to one complex POS-MORPH morphological output tag. Hence, we can model the position-wise probabilities $p(t|w_1^N)$ with recurrent neural networks, such as long short-term memory recurrent neural networks (LSTMs) [15]. Fig. 1 (a) shows such a network architecture where the inputs are the word vectors v_1^N. At the lower level, we use a CNN-based (Fig. 1 (b)) or an LSTM-based (Fig. 1 (c)) architecture to compute the character-based word vectors. As we are using bidirectional LSTMs (BLSTMs) at the top level, we shall refer to the complete architectures as CNNHighway-BLSTM and LSTM-BLSTM. The two architectures are fairly similar. In our opinion, however, there is an important difference between the two. CNNHighway is more constructive in the sense that it explicitly specifies the possible character context widths with a hard upper bound and defines an embedding size for each context width. LSTMs are more generic as they are claimed to implicitly learn these details [16].

The weights of the network, θ, are jointly estimated using conditional log-likelihood

$$F(\theta) = -\sum_{n=1}^{N} \log p_\theta(t_n|w_1^N). \qquad (2)$$

Learning in recurrent or very deep neural networks is non-trivial and skip/shortcut connections have been proposed to improve the learning of such networks [17], [18]. We use such connections (dashed arrows in Fig. 1) for LSTM-BLSTM to alleviate potential learning issues.

At test time, the predicted tag sequence is the tag sequence that maximizes the conditional probability $p(t_1^N|w_1^N)$. For the factorization in Eq. (1), the search can be done position-wise. This significantly reduces the computational and implementation complexity compared to first-order Markov models as used in [19], [1], [6].

III. Data

Most of the data sets are taken from the UD treebanks[1]. We also use a number of older data sets in order to compare our results with existing results in the literature, including Czech/PDT[2], German/TIGER[3], and Korean/SPMRL[4]. The corpus statistics for the different languages can be found in Table I. The training data are annotated with tags and used for supervised training. The Wiki dump data are not annotated and are only used to pre-computed word embeddings via word2vec [20]. The chosen languages are from different language families: Balto-Slavic (Bulgarian, Czech, Russian), Finnic (Estonian, Finnish), Finno-Ugric (Hungarian), Germanic (German), Indo-Iranian (Hindi), Koreanic (Korean), Romance (Romanian), Semitic (Arabic), and Turkic (Turkish). They include several examples for both agglutinative and fusional languages. The amount of supervised training data ranges from 33k training tokens (Hungarian/UD) to 1,174k training tokens (Czech/UD). The amount of unsupervised data used for pre-training word embeddings ranges from 3M training tokens (Arabic Wiki dump) to 2,252M training tokens (English Wiki dump), see last column of Table I.

Table II summarizes the tag statistics for the different languages. The number of tags is the number of POS-MORPH tags occurring in the training data. We give the test entropy based on a unigram tag model estimated on the training data as a simple measure for the difficulty of the associated sequence classification problem. The type/token ratio (TTR), also known as vocabulary size divided by text length, is computed on 1M words from randomly selected sentences from a different data set[5] and is a simple measure to quantify the morphological complexity of a language [21]. A higher TTR value indicates higher morphological complexity.

[1]http://dependencies.org/

[2]https://ufal.mff.cuni.cz/pdt3.0

[3]http://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger.html

[4]http://dokufarm.phil.hhu.de/spmrl2013/?animal=spmrl2013

[5]The sentences are all taken from the Wiki dumps on http://linguatools.org/tools/corpora/wikipedia-monolingual-corpora/ or https://archive.org/details/wikipediadumps?&sort=-downloads&page=2.

IV. Setups

We use the same model setups for LSTM-BLSTM and CNNHighway-BLSTM as in [14]. The hyper-parameters are set to

- CNNHighway: the large setup from [8], i.e., character vector size = 15, filter widths ranging from one to seven, number of filters as a function of the filter width $\min\{200, 50 \cdot \text{filter width}\}$, two highway layers
- LSTM: character vector size = 128, two layers with 1024 and 256 nodes

The BLSTM modeling the context of words in a sentence (Fig. 1 (a)) consists of two hidden layers, each with 256 hidden nodes. The pre-trained word vectors from word2vec [20] have the same as the character-based word vectors (256).

These hyper-parameters were tuned on the German TIGER development data and are optimal on a ”best effort basis” [14]. German is good for the hyper-parameter tuning as it is a relatively hard task (see Table II) and shows morphological effects both within and across words. Furthermore, the TIGER corpus is relatively large, which reduces statistical fluctuations in training and testing. Apart from these considerations, the choice was random. Furthermore, we tested language-specific tuning for a few languages, but it does not seem to give further gains. Moreover, the network hyper-parameters were tuned to give best accuracy rather than most compact models or even comparable numbers of parameters as our application is not constrained by memory or runtime. The hyper-parameters were then used for all languages. To establish state-of-the-art baselines, we ran the external tools MarMoT[6] and JNN[7] (see Appendix) with the suggested default values. MarMoT is based on conditional random fields using manually designed features [12]. JNN is a neural network toolkit and contains an LSTM-based tagger [2].

The networks are optimized as described in [14]. In particular, the optimization is done with RMSProp [22], with a fixed initial learning rate and a learning rate decay of two every tenth epoch for German/TIGER, and is adjusted for the other languages according to the amount of training data. The batch size is always 16. Furthermore, we use dropout. The dropout probability is empirically set to 0.4 for Hungarian and Turkish, which only have a very limited amount of training data (Table I), and to 0.2 for all other languages.

[6]http://cistern.cis.lmu.de/marmot/

[7]https://github.com/wlin12/JNN

Table I
CORPUS STATISTICS, OOV$\geq$5 DENOTES THE PERCENTAGE OF TEST WORD TOKENS WITH FIVE OR MORE OCCURRENCES IN THE TRAINING DATA

Language	Train sentences (k)	Train tokens (k)	Test tokens (k)	OOV$\geq$5 (%)	Wiki dump tokens (M)
Arabic/UD	6	256	32	20.7	3
Bulgarian/UD	9	124	16	27.3	46
Czech/PDT	39	691	93	17.5	83
UD	68	1174	174	15.7	83
English/UD	13	205	25	16.7	2252
Estonian/UD	15	188	24	32.2	21
Finnish/UD	12	163	9	38.9	64
French/UD	15	367	7	12.7	215
German/TIGER	40	760	92	17.2	610
Hindi/UD	13	281	35	10.1	32
Hungarian/UD	1	33	4	48.0	88
Korean/SPMRL	23	296	28	42.7	56
Romanian/UD	5	109	18	27.6	51
Russian/UD	47	815	108	19.5	68
Turkish/UD	4	42	9	46.5	49

Table II
TAG STATISTICS, TTR STANDS FOR TYPE/TOKEN RATIO

Language	#Tags	Entropy	TTR (%)
Arabic/UD	320	32.5	12
Bulgarian/UD	448	49.5	12
Czech/PDT	878	77.7	11
UD	1418	97.7	11
English/UD	119	27.9	7
Estonian/UD	787	57.3	13
Finnish/UD	1593	76.1	17
French/UD	197	34.1	8
German/TIGER	681	97.7	13
Hindi/UD	922	56.9	7
Hungarian/UD	652	64.5	14
Korean/SPMRL	1976	119.4	20
Romanian/UD	444	65.8	7
Russian/UD	434	54.6	16
Turkish/UD	987	73.0	10

V. EMPIRICAL EVALUATION

We empirically evaluate an LSTM-based and a CNN-based architecture for character-based morphological tagging (Section II) and compare them against MarMoT, a state-of-the-art morphological tagger [12], and JNN, a state-of-the-art part-of-speech tagger. For the evaluation we use twelve different morphologically-rich languages with different characteristics, plus two morphologically-poor languages for contrastive results (Section III). The configurations are described in Section IV.

Fig. 2 plots the relative gain over MarMoT (see Appendix-for more details) against the amount of training data. The horizontal dotted line at 0% indicates the MarMoT baseline. The blue squares are for LSTM-BLSTM results. Connecting them for the morphologically-rich languages shows a clear, nearly-linear dependency of the relative gain on the amount of training data. Only the data point for Turkish at 40% is an outlier (should be around 20%). This result suggests that compared to MarMoT, LSTM-BLSTM is very data efficient. Even for very small amounts of training data (e.g., 33k tokens for Hungarian), the relative gain is still 15%. On the other hand, more data helps. In case of Czech, increasing the amount of training data from 691k (Czech/PDT) to 1174k (Czech/UD) tokens leads to some additional gain and yields almost a 50% relative gain. It should be noted, however, that the two data sets use different tag sets, with the Czech/UD one being the more complex than the Czech/PDT (Table II).

We use an LSTM-BLSTM of the same size for all languages, although the amount of training data varies by roughly two orders of magnitude. Therefore, it is a valid question if a larger model specifically designed for Czech/UD or a smaller model for Turkish/UD would improve the results. We have developed carefully locally tuned and tested larger and smaller models in terms of number of nodes or layers but with similar or worse performance: -0.1% with more nodes (Czech) or approx. -1% with fewer nodes or fewer layers (Turkish). This observation suggests that the configuration optimized for German is fairly robust across many different languages, which is an attractive property from a practical perspective.

In contrast, we do not observe a gain of LSTM-BLSTM over MarMoT for English and French. Both languages are considered to be morphologically poor, as supported by the tag statistics in Table II. This may be because of the low morphological complexity, i.e., a character representation does not add much information to a word representation. Another explanation might be that the linguistic experts have focused on English and French in the last decades and found a good set of features, which however does not well generalize to other, morphologically more complex languages.

The red pluses are for the LSTM-BLSTM+word2vec results. LSTM-BLSTM+word2vec denotes an LSTM-BLSTM whose character-based word vector is concatenated with pre-trained word embeddings [20]. This allows us to exploit large amounts of unsupervised data in addition to the often limited amount of supervised data. The results (Fig. 2) are mixed but there is a clear trend that languages with smaller amounts of supervised training data benefit more from pre-trained word embeddings than languages with

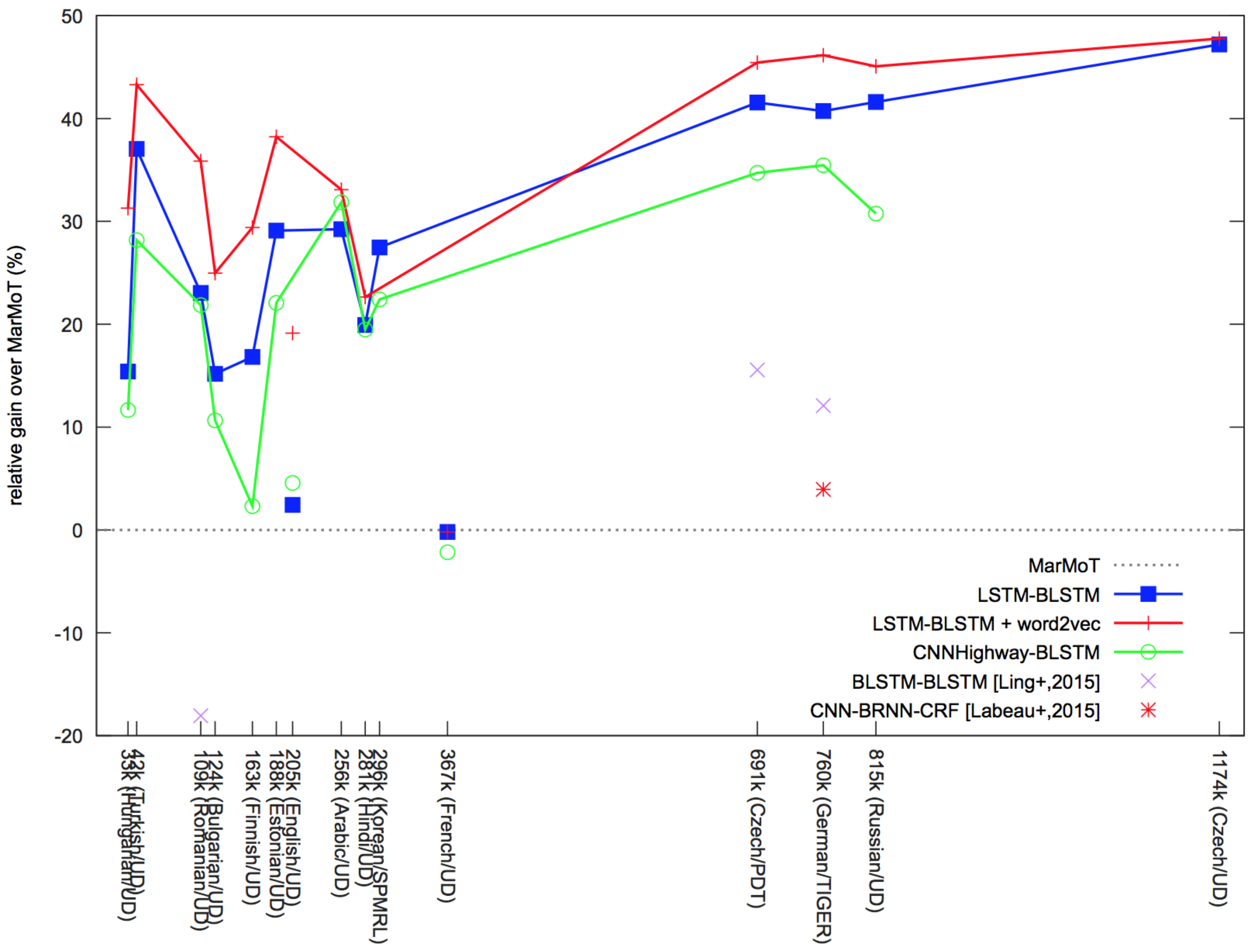

Figure 2. Relative gains (%) over MarMoT

larger amounts of supervised training data. For example, we observe an additional gain from 15% to over 30% for Hungarian/UD (supervised data with 33k training tokens only) but no additional gain at almost 50% for Czech/UD (supervised data with 1174k training tokens). These results suggest that pre-trained word embeddings may be a simple and effective way to close the performance gap between languages with small and large amounts of data.

It is tempting to analyze these results in more detail by splitting languages into sub-categories. Here, we refrain from doing so as it is delicate to draw conclusions from very small sample sizes (3-4 languages, say).

The green circles (in Fig. 2) are for CNNHighway-BLSTM results, a neural network architecture that has been developed for character-based language modeling [8]. Overall, LSTM-BLSTM and CNNHighway-BLSTM perform similarly, see Fig. 2. Looking at the details, however, CNNHighway-BLSTM tends to perform slightly worse and less consistently than LSTM-BLSTM.

For additional comparison, we add a few additional points in the plot. The single red cross indicates the result from [6], which is a combination of a CNN, a bidirectional RNN, and a Markov model. The purple crosses are generated with the external tool JNN[8], which implements a shallow BLSTM-BLSTM (i.e., only one bidirectional LSTM layer in each BLSTM). One might expect that this model performs better on smaller data sets. But actually, it is clearly worse both for large (Czech/PDT and German/TIGER) and small data sets (Romanian/UD).

We have not found CMC-based corpora for morphological tagging. To simulate noisy input data, we flip characters at random during training and testing as a simple preliminary experiment along these lines. Fig. 3 shows the results for German/TIGER. Adding noise to the clean model at test time, degrades the performance heavily, for example, from

[8]https://github.com/wlin12/JNN

less than 10% to almost 80% tag error rate for 20% character flips. Given an average word length of 6 characters, 20% character flips roughly correspond to one incorrect character per word. Using character flips for training lets the performance degrade much more gently. Moreover, the error rates only weakly depend on the noise level in training.

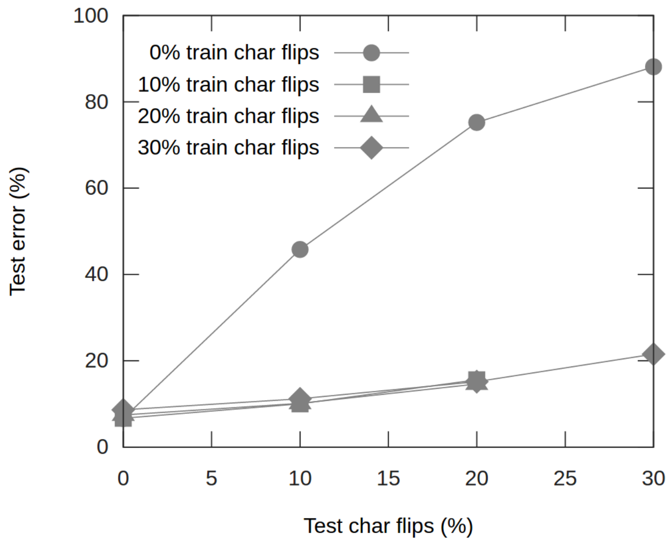

Figure 3. Effect of character flips on test error for German/TIGER corpus

To speed up the computation, the character-based word vectors of a sentence are processed in parallel. No further optimization has been done. This leads to a substantial speed up, in particular for LSTM-based word vectors. For a batch size of 16 and Titan X GPU, a training batch takes 0.7 seconds and 2.2 seconds for LSTM-BLSTM and HighwayCNN-BLSTM in our Torch7[9]-based implementation, respectively. HighwayCNN may be harder to parallelize because of the many small tensor operations. Doing only the forward propagation at test time, roughly halves the computation time.

VI. Summary & Future Work

In this paper, we demonstrated that a character-based neural approach can achieve consistent improvements over a state-of-the-art morphological tagger (MarMoT). The evaluation included a dozen of languages of different morphological complexity and with different characteristics. The relative gains for the morphologically-rich languages range from 15% to almost 50%, with a clear dependency on the amount of training data. Several aspects are remarkable about this result.

First, these results use the same model architecture with the same number of layers and nodes, without any language-specific modifications, Despite our best efforts, further local language and training data setting specific hyper-parameter tuning does not seem to result in performance gains.

Second, the neural approach seems to be more data efficient than the baseline tagger with manually designed features, also when only 30k training tokens are available.

Third, a fairly generic deep and hierarchical recurrent neural network architecture seems to perform as well or better than a more specialized convolutional neural network based architecture.

Fourth, to keep the setup as simple as possible initially, we have not used advanced techniques which are reported to lead to improvements, including additional unsupervised data (e.g., via pre-trained word vectors) [13], [2], [4], [5] and supplementary experiments in this work (see below), a non-trivial structured prediction model (e.g., a first-order Markov model) [19], [1], [6], [5], combination of different word representations [6], [5], [4], multilingual learning [3], [4], and auxiliary tasks [4]. Future work will include the investigation of these more advanced techniques. From this perspective, our paper provides a baseline for future research in multilingual character-based neural morphological tagging.

Last but not least, we do not observe any gains for English and French. This may be due to the low morphological complexity of these languages or because manual feature engineering has focused on these languages over the last decades with good results.

Moreover, we investigated the impact of using additional word embeddings trained on unlabeled supplementary data and how this is related to the amount of labeled data available. The results are somewhat mixed but we observe a trend that languages with smaller amounts of supervised training data benefit more from pre-trained word embeddings than languages with larger amounts of supervised training data. In particular, for English we did in fact observe performance increases using additional word embeddings trained on very large amounts of supplementary unlabeled data. For French we were not able to produce similar gains, perhaps also due to the smaller size of the supplementary unlabeled data, an order of magnitude less than what was available for English.

Finally, first experiments on noisy data (here, character flips to simulate typos) are very promising and we will extend our work along this line in the future.

Appendix

This appendix contains Table III with the raw results used in this paper. When available, the best comparable error rates from the literature are used. Otherwise, we produced the error rates with the publicly available tools and the suggested default values. More specifically, we used the state-of-the-art tagger MarMoT[10] for the baselines and the LSTM-based POS tagger JNN[11] for some contrastive results.

[9]http://torch.ch/

[10]http://cistern.cis.lmu.de/marmot/

[11]https://github.com/wlin12/JNN

Table III
TAG ERROR RATES (%) ON TEST SETS, SOME OF WHICH ARE TAKEN FROM THE LITERATURE: (A) [12], (B) [6]

Language	MarMoT[10]	CNN-biRNN-CRF	BLSTM-BLSTM[11]	CNNHighway-BLSTM	LSTM-BLSTM	LSTM-BLSTM+word2vec
Arabic/UD	9.13			6.22	6.46	6.11
Bulgarian/UD	5.73			5.12	4.86	4.30
Czech/PDT	7.46[a]		6.30	4.87	4.36	4.07
UD	6.97				3.68	3.64
English/UD	7.00			6.68	6.83	5.66
Estonian/UD	8.11			6.32	5.75	5.01
Finnish/UD	7.79			7.61	6.48	5.50
French/UD	5.08			5.19	5.09	5.09
German/TIGER	11.42[a]	10.97[b]	10.04	7.37	6.77	6.15
Hindi/UD	11.44			9.21	9.16	8.85
Hungarian/UD	26.49			23.40	22.41	18.20
Korean/SPMRL	18.60			14.43	13.49	
Romanian/UD	7.64		9.02	5.97	5.88	4.90
Russian/UD	6.08			4.21	3.55	3.34
Turkish/UD	17.28			12.41	10.88	9.80

ACKNOWLEDGMENT

This work has been partly funded by the European Unions Horizon 2020 research and innovation programme under grant agreement No. 645452 (QT21).

REFERENCES

[1] C. dos Santos and B. Zadrozny, "Learning character-level representations for part-of-speech tagging," in *ICML*, Beijing, China, Jun. 2014.

[2] W. Ling, T. Luís, L. Marujo, R. F. Astudillo, S. Amir, C. Dyer, A. Black, and I. Trancoso, "Finding function in form: Compositional character models for open vocabulary word representation," in *EMNLP*, Lisbon, Portugal, Sep. 2015.

[3] D. Gillick, C. Brunk, O. Vinyals, and A. Subramanya, "Multilingual language processing from bytes," Dec. 2015.

[4] B. Plank, A. Søgaard, and Y. Goldberg, "Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss," in *ACL*, Berlin, Germany, Aug. 2016.

[5] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF," in *ACL*, Berlin, Germany, Aug. 2016.

[6] M. Labeau, K. Löser, and A. Allauzen, "Non-lexical neural architecture for fine-grained POS tagging," in *EMNLP*, Lisbon, Portugal, Sep. 2015.

[7] M. Ballesteros, C. Dyer, and N. Smith, "Improved transition-based parsing by modeling characters instead of words with LSTMs," in *EMNLP*, Lisbon, Portugal, Sep. 2015.

[8] Y. Kim, Y. Jernite, D. Sontag, and A. Rush, "Character-aware neural language models," in *AAAI*, Phoenix, AZ, USA, Feb. 2016.

[9] M. Costa-jussà and J. Fonollosa, "Character-based neural machine translation," in *ACL*, Berlin, Germany, Aug. 2016.

[10] K. Oflazer and I. Kuroz, "Tagging and morphological disambiguation of Turkish text," in *Proceedings of the Applied natural language processing*, 1994.

[11] J. Hajič and B. Hladk, "Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset," in *Proceedings of Coling*, 1998.

[12] T. Müller, H. Schmid, and H. Schütze, "Efficient higher-order CRFs for morphological tagging," in *ACL*, Seattle, WA, USA, Oct. 2013.

[13] T. Müller and H. Schütze, "Robust morphological tagging with word representations," in *ACL*, Denver, CO, USA, Jun. 2015.

[14] G. Heigold, G. Neumann, and J. van Genabith, "Neural morphological tagging from characters for morphologically rich languages," *CoRR*, vol. abs/1606.06640, 2016. [Online]. Available: http://arxiv.org/abs/1606.06640

[15] A. Graves, *Supervised sequence labelling with recurrent neural networks*, ser. Studies in Computational Intelligence. Heidelberg, New York: Springer, 2012. [Online]. Available: http://opac.inria.fr/record=b1133792

[16] J. Schmidhuber, "Learning complex, extended sequences using the principle of history compression," *Neural Computation*, vol. 4, no. 2, pp. 234–242, 1992.

[17] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," in *ICLR*, 2014.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[19] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[20] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, Lake Tahoe, CA, USA, Dec. 2013.

[21] M. Bane, "Quantifying and measuring morphological complexity," in *Proceedings of the 26th West Coast Conference on Formal Linguistics*, C. Chang and H. Haynie, Eds. Somerville, MA, USA: Cascadilla Proceedings Project, 2008, pp. 69–76.

[22] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA: Neural Networks for Machine Learning*, vol. 4, 2012.