Non-lexical neural architecture for fine-grained POS Tagging

Matthieu Labeau, Kevin Löser, Alexandre Allauzen
Université Paris-Sud and LIMSI-CNRS,
Rue John von Neumann
91403 Orsay cedex
France
firstname.lastname@limsi.fr

Abstract

In this paper we explore a POS tagging application of neural architectures that can infer word representations from the raw character stream. It relies on two modelling stages that are jointly learnt: a convolutional network that infers a word representation directly from the character stream, followed by a prediction stage. Models are evaluated on a POS and morphological tagging task for German. Experimental results show that the convolutional network can infer meaningful word representations, while for the prediction stage, a well designed and structured strategy allows the model to outperform state-of-the-art results, without any feature engineering.

1 Introduction

Most modern statistical models for natural language processing (NLP) applications are strongly or fully lexicalized, for instance part-of-speech (POS) and named entity taggers, as well as language models, and parsers. In these models, the observed word form is considered as the elementary unit, while its morphological properties remain neglected. As a result, the vocabulary observed on training data heavily restricts the generalization power of lexicalized models.

Designing subword-level systems is appealing for several reasons. First, words sharing morphological properties often share grammatical function and meaning, and leveraging that information can yield improved word representations. Second, a subword-level analysis can address the out-of-vocabulary issue *i.e* the fact that word-level models fail to meaningfully process unseen word forms. This allows a better processing of morphologically rich languages in which there is a combinatorial explosion of word forms, most of which are not observed during training. Finally, using subword units could allow processing of noisy text such as user-generated content on the Web, where abbreviations, slang usage and spelling mistakes cause the number of word types to explode.

This work investigates models that do not rely on a fixed vocabulary to make a linguistic prediction. Our main focus in this paper is POS tagging, yet the proposed approach could be applied to a wide variety of language processing tasks. Our main contribution is to show that neural networks can successfully learn unlexicalized models that infer a useful word representation from the character stream. This approach achieves state of-the-art performance on a German POS tagging task. This task is difficult because German is a morphologically rich language[1], as reflected by the large number of morphological tags (255) in our study, yielding a grand total of more than 600 POS+MORPH tags. An aggravating factor is that these morphological categories are overtly marked by a handful of highly ambiguous inflection marks (suffixes). We therefore believe that this case study is well suited to assess both the representation and prediction power of our models.

The architecture we explore in section 2 differs from previous work that only consider the character level. Following (Santos and Zadrozny, 2014), it consists in two stages that are jointly learnt. The lower stage is a convolutional network that infers a word embedding from a character string of arbitrary size, while the higher network infers the POS tags based on this word embedding sequence. For the latter, we investigate different architectures of increasing complexities: from a feedforward and context-free inference to a bi-recurrent network that predicts the global sequence. Experimental results (section 4) show that the proposed approach can achieve state of the art performance

[1]Besides inflected forms, German is characterized by a possibly infinite and evolving set of compound nouns.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 232–237,
Lisbon, Portugal, 17-21 September 2015.

and that the choice of architecture for the prediction part of the model has a significant impact.

2 Network Architectures

The different architectures we propose act in two stages to infer, for a sentence $s = \{w_1, \ldots, w_{|s|}\}$, a sequence of tags $\{t_1, \ldots, t_{|s|}\}$. Each tag belongs to the tagset $\mathcal{T}$. The first stage is designed to represent each word *locally*, and focuses on capturing the meaningful morphological information. In the second stage, we investigate different ways to predict the tag sequence that differ in how the *global* information is used.

2.1 From character to word level

To obtain word embeddings, the usual approach introduced by (Bengio et al., 2003) relies on a fixed vocabulary $\mathcal{W}$ and each word $w \in \mathcal{W}$ is mapped to a vector of n_f real valued features by a look-up matrix $W \in \mathbb{R}^{|\mathcal{W}|*n_f}$. To avoid the use of a fixed vocabulary, we propose to derive a word representation from a sequence of character embedding: if $\mathcal{C}$ denotes the finite set of characters, each character is mapped on a vector of n_c features gathered in the look-up matrix C.

To infer a word embedding , we use a *convolution layer* (Waibel et al., 1990; Collobert et al., 2011), build as in (Santos and Zadrozny, 2014). As illustrated in figure 1, a word w is a character sequence $\{c_1, .., c_{|w|}\}$ represented by their embeddings $\{C_{c_1}, .., C_{c_{|w|}}\}$, where C_{c_i} denotes the row in C associated to the character c_i. A convolution filter $W^{conv} \in \mathbb{R}^{n_f} \times \mathbb{R}^{d_c * n_c}$ is applied over a sliding window of d_c characters, producing local features :

$$x_n = W^{conv}(C_{c_{n-d_c+1}} : .. : C_{c_n})^T + b^{conv},$$

where x_n is a vector of size n_f obtained for each position n in the word[2]. The i-th element of the embedding of w is the maximum over the i-th elements of the feature vectors :

$$[f]_i = \tanh(\max_{1 \leq n \leq |s|} [x_n]_i)$$

Using a maximum after a sliding convolution window ensures that the embedding combines local features from the whole word, and selects the more useful ones. The parameters of the layer are the matrices C and W^{conv} and the bias b^{conv}.

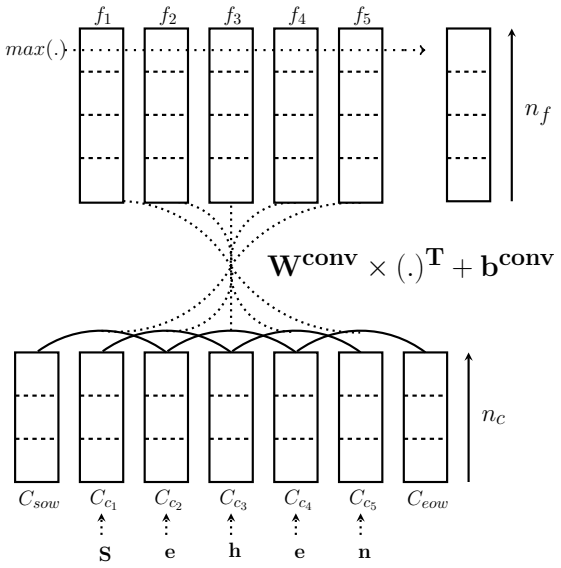

Figure 1: Architecture of the layer for character-level encoding of words.

2.2 From words to prediction

To predict the tag sequence associated to a sentence s, we first use **a feedforward architecture**, with a single hidden layer. To compute the probability of tagging the n-th word in the sentence with tag t_i, we use a window of d_w word embeddings[3] centered around the word w_n:

$$x_n = f_{n-\frac{d_w-1}{2}} : ... : f_{n+\frac{d_w-1}{2}},$$

followed by a hidden and output layers:

$$s_n = W^o \tanh(W^h x_n + b^h) + b^o. \qquad (1)$$

The parameters of the hidden an output layers are respectively W^h, b^h and W^o, b^o.

We also experiment with a **a bidirectional recurrent layer**, as described in (Graves et al., 2013). The forward and backward passes allow each prediction to be conditioned on the complete past and future contexts, instead of merely a neighboring window. As illustrated in figure 2, the forward hidden state, at position n, will be computed using the previous forward hidden state and the word embedding in position n:

$$\overrightarrow{h^n} = \tanh(\overrightarrow{W^{fh}} f_n + \overrightarrow{W^{hh}} \overrightarrow{h^{n-1}} + b^h)$$

[2]Two padding character tokens are used to deal with border effects. The first is added at the beginning and the second at the end of the word, as many times as it is necessary to obtain the same number of windows than the length of the word. Their embeddings are added to C.

[3]Similarly, we use special word tokens for padding.

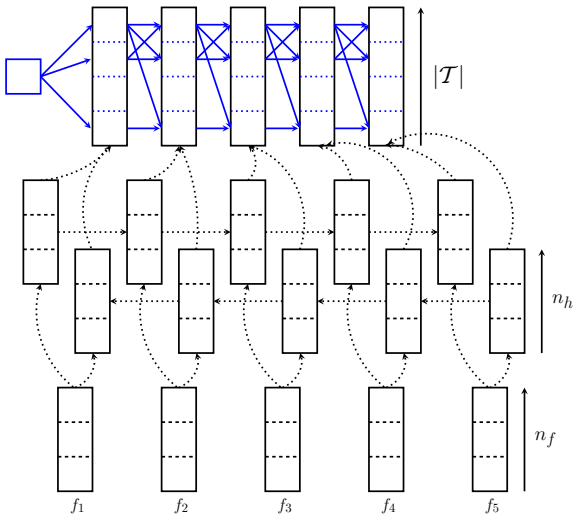

Figure 2: Bidirectional recurrent architecture for tag prediction. The upper part is used in the case of structured inference.

$\overrightarrow{W^{fh}}$ and $\overrightarrow{W^{hh}}$ are the transition matrices of the forward part of the layer, and b_h is the bias. The backward hidden states are computed similarly, and the hidden states of each direction are concatenated to pass through an output layer:

$$s_n = W^o(\overrightarrow{h^n} : \overleftarrow{h^n}) + b^o. \quad (2)$$

2.3 Inference and Training

To infer the tag sequence from the sequence of output layers defined by equations 1 or 2, we explore two strategies. The first simply applies a softmax function to the output layer of the network described in the previous section. In this case, each tag prediction is made independently of the surrounding predictions.

For sequence labeling, a more appropriate solution relies on the approach of (Collobert, 2011), also used in (Santos and Zadrozny, 2014). Let consider each possible tag sequence $\{t_1, \ldots, t_{|s|}\}$ as a possible path over a sequence of hidden states. We can add a transition matrix W^{trans} and then compute the score of a sequence as follows:

$$s(\{t\}_1^{|s|}, \{w\}_1^{|s|}) = \sum_{1 \leq n \leq |s|} \left(W^{trans}_{t_{n-1}, t_n} + [s_n]_{t_n} \right)$$

The Viterbi algorithm (Viterbi, 1967) offers an exact solution to infer the path that gives the maximum score. It is worth noticing that both these strategies can be applied to the feedforward and bidirectional recurrent networks. For both strategies, the whole network can estimate conditional log-likelihood of a tag sequence given a sentence s and the set of parameters θ. This criterion can then be optimized using a stochastic gradient ascent with the back-propagation algorithm.

3 Related Work

The choice to consider words from the character level has recently been more and more explored. While its raw application to language modeling did not achieve clear improvement over the word-based models (Mikolov et al., 2012), this approach shown impressive results for text generation (Sutskever et al., 2011; Graves, 2013). However, for this line of work, the main issue is to learn long range dependencies at the character level since the word level is not considered by the model.

More recently, the character level was considered as more interpretable and convenient way to explore and understand recurrent networks (Karpathy et al., 2015). In (Zhang and LeCun, 2015), the authors build a text understanding model that does not require any knowledge and uses hierarchical feature extraction. Here the character level allows the model to ignore the definition *a priori* of a vocabulary and let the model build its own representation of a sentence or a document, directly from the character level. To some extent, our work can be considered as an extension of their work, tailored for POS tagging.

(Santos and Zadrozny, 2014) applies a very similar model to the POS tagging of Portuguese and English. (Luong et al., 2013) also descends lower than the word level, using a dictionary of morphemes and recursive neural networks to model the structure of the words. Similarly, this allows a better representation of rare and complex words, evaluated on a word similarity task.

4 Experiments and Results

Experiments are carried out on the Part-of-Speech and Morphological tagging tasks using the German corpus TIGER Treebank (Brants et al., 2002). To the best of our knowledge, the best results on this task were published in (Mueller et al., 2013), who applied a high-order CRF that includes an intensive feature engineering to five different languages. German was highlighted as having 'the most ambiguous morphology'. The corpus, de-

Architecture	Encoding	Output	POS		POS+Morph	
			Dev	Test	Dev	Test
Feedforward	Lex.	Simple	4.22 ± 0.05	5.89 ± 0.07	13.97 ± 0.14	17.46 ± 0.14
		Struct.	3.90 ± 0.05	5.33 ± 0.09	12.22 ± 0.13	15.34 ± 0.13
	Non-lex.	Simple	3.31 ± 0.07	4.22 ± 0.07	13.50 ± 0.16	16.23 ± 0.13
		Struct.	2.92 ± 0.02	3.82 ± 0.04	11.65 ± 0.11	14.43 ± 0.19
	Both	Simple	2.59 ± 0.05	3.34 ± 0.09	11.89 ± 0.14	14.63 ± 0.22
		Struct.	$2.22 \pm 0.03^*$	$2.86 \pm 0.03^*$	9.11 ± 0.14	11.29 ± 0.06
biRNN	Lex	Simple	6.03 ± 0.06	8.05 ± 0.05	17.83 ± 0.11	21.33 ± 0.26
		Struct.	3.89 ± 0.06	5.26 ± 0.05	11.88 ± 0.05	17.78 ± 0.12
	Non-Lex	Simple	4.46 ± 0.08	5.84 ± 0.19	16.61 ± 0.18	19.39 ± 0.12
		Struct.	2.74 ± 0.07	3.59 ± 0.07	10.09 ± 0.09	12.88 ± 0.28
	Both	Simple	3.63 ± 0.06	4.63 ± 0.04	14.83 ± 0.11	17.54 ± 0.13
		Struct.	$2.21 \pm 0.04^*$	$2.86 \pm 0.05^*$	$\mathbf{8.63 \pm 0.21}^*$	$\mathbf{10.97 \pm 0.19}^*$
	CRF		**2.06**	**2.56**	9.40	11.42

Table 1: Comparison of the feedforward and bidirectional recurrent architectures for predictions, with different settings. The non-lexical encoding is convolutional. *CRF* refers to state-of-the-art system of (Mueller et al., 2013). *Simple* and *Struct.* respectively denote the position-by-position and structured prediction. $*$ indicates our best configuration.

scribed in details in (Fraser et al., 2013), contains a training set of 40472 sentences, a development and a test set of both 5000 sentences. We consider the two tagging tasks, with first a coarse tagset (54 tags), and then a morpho-syntactical rich tagset (619 items observed on the the training set).

4.1 Experimental settings

All the models are implemented[4] with the Theano library (Bergstra et al., 2010). For optimization, we use Adagrad (Duchi et al., 2011), with a learning rate of 0.1. The other hyperparameters are: the window sizes, d_c and d_w, respectively set to 5 and 9, the dimension of character embeddings, word embeddings and of the hidden layer, n_c, n_f and n_h, that are respectively of 100, 200 and 200[5]. The models were trained on 7 epochs. Parameter initialization and corpus ordering are random, and the results presented are the average and standard deviation of the POS Tagging error rate over 5 runs.

[4]Implementation is available at `https://github.com/MatthieuLabeau/NonlexNN`

[5]For both the learning rate and the embedding sizes, results does not differ in a significant way in a large range of hyperparameters, and their impact resides more in convergence speed and computation time

4.2 Results

The first experiment aims to evaluate the efficiency of a convolutional encoding with the basic feedforward architecture for prediction. We compare a completely non-lexicalized model which relies only on a character-level encoding with a lexicalized model where we use conventional word embeddings stored with a fixed vocabulary[6]. Results are reported in Table 1 along with with the state-of-the-art results published in (Mueller et al., 2013). Results show that a character-level encoding yields better results than the conventional word-level encoding. Moreover, the structured inference allows the model to achieve accuracy reasonably close to the performance of a high-order CRF that uses handcrafted features. Finally, the model that uses the concatenation of both the character and word-level embeddings outperforms the state-of-the-art system on the more difficult task, without any feature engineering.

To give an idea of how a simple model would perform on such task, the reader can refer to (Schmid and Laws, 2008) and (Mueller et al., 2013). For instance in the former, by choosing the most probable tag position-by-position, the error rate on the development set of the TIGER dataset

[6]Every word that appears in the training set.

is 32.7 for the simple POS Tagging task.

We further analyze the results by looking at the error rates respectively on known and unknown words[7]. From table 2, we observe that the number of unknown words wrongly labeled is divided by 3 for POS and almost divided by 2 for POS+Morph tagging, showing the ability of character-level encoding to generalize to new words. Moreover, a strictly non-lexical encoding makes slightly more mistakes on words already seen, whereas the model that concatenates both embeddings will make less mistakes for both unknown and known words.

This shows that information from the context and from the morphology are complementary, which is conjectured in (Mueller et al., 2013) by using a morphological analyzer in complement of higher-order CRF.

		Lex.	Non-lex.	Both
POS	Unknown	2970	1054	1010
	Known	1974	2981	1620
POS+Morph	Unknown	5827	3472	3384
	Known	8652	10205	7232

Table 2: Error counts for known/unknown words in the test set, with a structured feedforward prediction model for the tagging task.

In the second set of experiments, we evaluate the convolutional encoding with a bidirectional recurrent network for prediction. Results are presented in the second half of Table 1. Surprisingly, this architecture performs poorly with simple inference, but clearly improves when predicting a structured output using the Viterbi algorithm, both for training and testing. Moreover, a non-lexical model trained to infer a tag sequence with the Viterbi algorithm achieves results that are close to the state-of-the-art, thus validating our approach. We consider that this improvement comes from the synergy between using a global training objective with a global hidden representation, complexifying the model but allowing a more efficient solution. Finally, the model that uses the combination of both the character and word-level embeddings yields the best results. It is interesting to notice that the predictive architecture has no influence on the results of the simple task when the prediction is structured, but improves them on the difficult task. This also shows that the contribution of word embeddings to our model corresponds to a difference of 1.5 to 2 points in performance.

[7]Unknown words refer to words present in the development or test sets, but not in the training set.

5 Conclusion

In this paper, we explored new models that can infer meaningful word representations from the raw character stream, allowing the model to exploit the morphological properties of words without using any handcrafted features or external tools. These models can therefore efficiently process words that were unseen in the training data. The evaluation was carried out on a POS and morphological tagging task for German. We described different architectures that act in two stages: the first stage is a convolutional network that infers a word representation directly from the character stream, while the second stage performs the prediction. For the prediction stage, we investigated different solutions showing that a bidirectional recurrent network can outperform state-of-the-art results when using a structured inference algorithm.

Our results showed that character-level encoding can address the unknown words problem for morphologically complex languages. In the future, we plan to extend these models to other tasks such as syntactic parsing and machine translation. Moreover, we will also investigate other architectures to infer word embeddings from the character level. For instance, preliminary experiments show that bidirectional recurrent network can achieve very competitive and promising results.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments and suggestions. This work has been partly funded by the European Unions Horizon 2020 research and innovation programme under grant agreement No. 645452 (QT21).

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137 1155.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and

GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June. Oral Presentation.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, pages 24–41.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 224–232.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July.

Alexander Fraser, Helmut Schmid, Richárd Farkas, Renjing Wang, and Hinrich Schütze. 2013. Knowledge sources for constituent parsing of German, a morphologically rich and less-configurational language. *Comput. Linguist.*, 39(1):57–85, March.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, December 8-12, 2013*, pages 273–278.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078.

Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013, Sofia, Bulgaria, August 8-9, 2013*, pages 104–113.

Tomas Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. 2012. Subword language modeling with neural networks. Unpublished.

Thomas Mueller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA, October. Association for Computational Linguistics.

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

Helmut Schmid and Florian Laws. 2008. Estimation of conditional probabilities with decision trees and an application to fine-grained pos tagging. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 777–784, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 1017–1024, New York, NY, USA, June. ACM.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theor.*, 13(2):260–269, April.

Alexander Waibel, Toshiyuki Hanazawa, Geofrey Hinton, Kiyohiro Shikano, and Kevin J. Lang, 1990. *Readings in Speech Recognition*, chapter Phoneme Recognition Using Time-delay Neural Networks, pages 393–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *CoRR*, abs/1502.01710.