Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss

Barbara Plank
University of Groningen
The Netherlands
b.plank@rug.nl

Anders Søgaard
University of Copenhagen
Denmark
soegaard@hum.ku.dk

Yoav Goldberg
Bar-Ilan University
Israel
yoav.goldberg@gmail.com

Abstract

Bidirectional long short-term memory (bi-LSTM) networks have recently proven successful for various NLP sequence modeling tasks, but little is known about their reliance to input representations, target languages, data set size, and label noise. We address these issues and evaluate bi-LSTMs with word, character, and unicode byte embeddings for POS tagging. We compare bi-LSTMs to traditional POS taggers across languages and data sizes. We also present a novel bi-LSTM model, which combines the POS tagging loss function with an auxiliary loss function that accounts for rare words. The model obtains state-of-the-art performance across 22 languages, and works especially well for morphologically complex languages. Our analysis suggests that bi-LSTMs are less sensitive to training data size and label corruptions (at small noise levels) than previously assumed.

1 Introduction

Recently, bidirectional long short-term memory networks (bi-LSTM) (Graves and Schmidhuber, 2005; Hochreiter and Schmidhuber, 1997) have been used for language modelling (Ling et al., 2015), POS tagging (Ling et al., 2015; Wang et al., 2015), transition-based dependency parsing (Ballesteros et al., 2015; Kiperwasser and Goldberg, 2016), fine-grained sentiment analysis (Liu et al., 2015), syntactic chunking (Huang et al., 2015), and semantic role labeling (Zhou and Xu, 2015). LSTMs are recurrent neural networks (RNNs) in which layers are designed to prevent vanishing gradients. Bidirectional LSTMs make a backward and forward pass through the sequence before passing on to the next layer. For further details, see (Goldberg, 2015; Cho, 2015).

We consider using bi-LSTMs for POS tagging. Previous work on using deep learning-based methods for POS tagging has focused either on a single language (Collobert et al., 2011; Wang et al., 2015) or a small set of languages (Ling et al., 2015; Santos and Zadrozny, 2014). Instead we evaluate our models across 22 languages. In addition, we compare performance with representations at different levels of granularity (words, characters, and bytes). These levels of representation were previously introduced in different efforts (Chrupała, 2013; Zhang et al., 2015; Ling et al., 2015; Santos and Zadrozny, 2014; Gillick et al., 2016; Kim et al., 2015), but a comparative evaluation was missing.

Moreover, deep networks are often said to require large volumes of training data. We investigate to what extent bi-LSTMs are more sensitive to the amount of training data and label noise than standard POS taggers.

Finally, we introduce a novel model, a bi-LSTM trained with auxiliary loss. The model jointly predicts the POS and the log frequency of the next word. The intuition behind this model is that the auxiliary loss, being predictive of word frequency, helps to differentiate the representations of rare and common words. We indeed observe performance gains on rare and out-of-vocabulary words. These performance gains transfer into general improvements for morphologically rich languages.

Contributions In this paper, we a) evaluate the effectiveness of different representations in bi-LSTMs, b) compare these models across a large set of languages and under varying conditions (data size, label noise) and c) propose a novel bi-LSTM model with auxiliary loss (LOGFREQ).

2 Tagging with bi-LSTMs

Recurrent neural networks (RNNs) (Elman, 1990) allow the computation of fixed-size vector representations for word sequences of arbitrary length. An RNN is a function that reads in n vectors $x_1, ..., x_n$ and produces an output vector h_n, that depends on the entire sequence $x_1, ..., x_n$. The vector h_n is then fed as an input to some classifier, or higher-level RNNs in stacked/hierarchical models. The entire network is trained jointly such that the hidden representation captures the important information from the sequence for the prediction task.

A bidirectional recurrent neural network (bi-RNN) (Graves and Schmidhuber, 2005) is an extension of an RNN that reads the input sequence twice, from left to right and right to left, and the encodings are concatenated. The literature uses the term bi-RNN to refer to two related architectures, which we refer to here as "context bi-RNN" and "sequence bi-RNN". In a sequence bi-RNN (bi-RNN_{seq}), the input is a sequence of vectors $x_{1:n}$ and the output is a concatenation ($\circ$) of a forward (f) and reverse (r) RNN each reading the sequence in a different directions:

$$v = \text{bi-RNN}_{\text{seq}}(x_{1:n}) = \text{RNN}_f(x_{1:n}) \circ \text{RNN}_r(x_{n:1})$$

In a context bi-RNN (bi-RNN_{ctx}), we get an additional input i indicating a sequence position, and the resulting vectors v_i result from concatenating the RNN encodings up to i:

$$v_i = \text{bi-RNN}_{\text{ctx}}(x_{1:n}, i) = \text{RNN}_f(x_{1:i}) \circ \text{RNN}_r(x_{n:i})$$

Thus, the state vector v_i in this bi-RNN encodes information at position i and its entire sequential context. Another view of the context bi-RNN is of taking a sequence $x_{1:n}$ and returning the corresponding sequence of state vectors $v_{1:n}$.

LSTMs (Hochreiter and Schmidhuber, 1997) are a variant of RNNs that replace the cells of RNNs with LSTM cells that were designed to prevent vanishing gradients. Bidirectional LSTMs are the bi-RNN counterpart based on LSTMs.

Our basic bi-LSTM tagging model is a context bi-LSTM taking as input word embeddings $\vec{w}$. We incorporate subtoken information using an hierarchical bi-LSTM architecture (Ling et al., 2015; Ballesteros et al., 2015). We compute subtoken-level (either characters $\vec{c}$ or unicode byte $\vec{b}$) embeddings of words using a sequence bi-LSTM at the lower level. This representation is then concatenated with the (learned) word embeddings vector $\vec{w}$ which forms the input to the context bi-LSTM at the next layer. This model, illustrated in Figure 1 (lower part in left figure), is inspired by Ballesteros et al. (2015). We also test models in which we only keep sub-token information, e.g., either both byte and character embeddings (Figure 1, right) or a single (sub-)token representation alone.

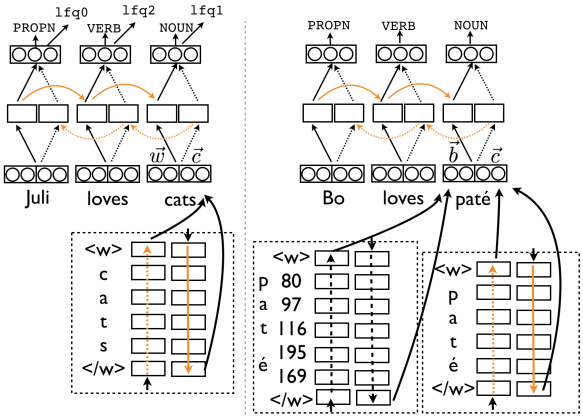

Figure 1: Right: bi-LSTM, illustrated with $\vec{b} + \vec{c}$ (bytes and characters), for $\vec{w} + \vec{c}$ replace $\vec{b}$ with words $\vec{w}$. Left: FREQBIN, our multi-task bi-LSTM that predicts at every time step the tag and the frequency class for the next token.

In our novel model, cf. Figure 1 left, we train the bi-LSTM tagger to predict both the tags of the sequence, as well as a label that represents the log frequency of the next token as estimated from the training data. Our combined cross-entropy loss is now: $L(\hat{y}_t, y_t) + L(\hat{y}_a, y_a)$, where t stands for a POS tag and a is the log frequency label, i.e., $a = int(log(freq_{train}(w))$. Combining this log frequency objective with the tagging task can be seen as an instance of multi-task learning in which the labels are predicted jointly. The idea behind this model is to make the representation predictive for frequency, which encourages the model to not share representations between common and rare words, thus benefiting the handling of rare tokens.

3 Experiments

All bi-LSTM models were implemented in CNN/pycnn,[1] a flexible neural network library. For all models we use the same hyperparameters, which were set on English dev, i.e., SGD training with cross-entropy loss, no mini-batches, 20

[1] https://github.com/clab/cnn

epochs, default learning rate (0.1), 128 dimensions for word embeddings, 100 for character and byte embeddings, 100 hidden states and Gaussian noise with σ=0.2. As training is stochastic in nature, we use a fixed seed throughout. Embeddings are not initialized with pre-trained embeddings, except when reported otherwise. In that case we use off-the-shelf `polyglot` embeddings (Al-Rfou et al., 2013).[2] No further unlabeled data is considered in this paper. The code is released at: `https://github.com/bplank/bilstm-aux`

Taggers We want to compare POS taggers under varying conditions. We hence use three different types of taggers: our implementation of a bi-LSTM; TNT (Brants, 2000)—a second order HMM with suffix trie handling for OOVs. We use TNT as it was among the best performing taggers evaluated in Horsmann et al. (2015).[3] We complement the NN-based and HMM-based tagger with a CRF tagger, using a freely available implementation (Plank et al., 2014) based on `crfsuite`.

3.1 Datasets

For the multilingual experiments, we use the data from the Universal Dependencies project v1.2 (Nivre et al., 2015) (17 POS) with the canonical data splits. For languages with token segmentation ambiguity we use the provided gold segmentation. If there is more than one treebank per language, we use the treebank that has the canonical language name (e.g., *Finnish* instead of *Finnish-FTB*). We consider all languages that have at least 60k tokens and are distributed with word forms, resulting in 22 languages. We also report accuracies on WSJ (45 POS) using the standard splits (Collins, 2002; Manning, 2011). The overview of languages is provided in Table 1.

3.2 Results

Our results are given in Table 2. First of all, notice that TNT performs remarkably well across the 22 languages, closely followed by CRF. The bi-LSTM tagger ($\vec{w}$) without lower-level bi-LSTM for subtokens falls short, outperforms the traditional taggers only on 3 languages. The bi-LSTM model clearly benefits from character representations. The model using characters alone ($\vec{c}$) works remarkably well, it improves over TNT on 9 languages (incl. Slavic and Nordic languages). The combined word+character representation model is the best representation, outperforming the baseline on all except one language (Indonesian), providing strong results already without pre-trained embeddings. This model ($\vec{w} + \vec{c}$) reaches the biggest improvement (more than +2% accuracy) on Hebrew and Slovene. Initializing the word embeddings (+POLYGLOT) with off-the-shelf language-specific embeddings further improves accuracy. The only system we are aware of that evaluates on UD is Gillick et al. (2016) (last column). However, note that these results are not strictly comparable as they use the earlier UD v1.1 version.

	COARSE	FINE		COARSE	FINE
ar	non-IE	Semitic	he	non-IE	Semitic
bg	Indoeuropean	Slavic	hi	Indoeuropean	Indo-Iranian
cs	Indoeuropean	Slavic	hr	Indoeuropean	Slavic
da	Indoeuropean	Germanic	id	non-IE	Austronesian
de	Indoeuropean	Germanic	it	Indoeuropean	Romance
en	Indoeuropean	Germanic	nl	Indoeuropean	Germanic
es	Indoeuropean	Romance	no	Indoeuropean	Germanic
eu	Language isolate		pl	Indoeuropean	Slavic
fa	Indoeuropean	Indo-Iranian	pt	Indoeuropean	Romance
fi	non-IE	Uralic	sl	Indoeuropean	Slavic
fr	Indoeuropean	Romance	sv	Indoeuropean	Germanic

Table 1: Grouping of languages.

The overall best system is the multi-task bi-LSTM FREQBIN (it uses $\vec{w} + \vec{c}$ and POLYGLOT initialization for $\vec{w}$). While on macro average it is on par with bi-LSTM $\vec{w} + \vec{c}$, it obtains the best results on 12/22 languages, and it is successful in predicting POS for OOV tokens (cf. Table 2 OOV ACC columns), especially for languages like Arabic, Farsi, Hebrew, Finnish.

We examined simple RNNs and confirm the finding of Ling et al. (2015) that they performed worse than their LSTM counterparts. Finally, the bi-LSTM tagger is competitive on WSJ, cf. Table 3.

Rare words In order to evaluate the effect of modeling sub-token information, we examine accuracy rates at different frequency rates. Figure 2 shows absolute improvements in accuracy of bi-LSTM $\vec{w} + \vec{c}$ over mean log frequency, for different language families. We see that especially for Slavic and non-Indoeuropean languages, having high morphologic complexity, most of the improvement is obtained in the Zipfian tail. Rare tokens benefit from the sub-token representations.

[2] `https://sites.google.com/site/rmyeid/projects/polyglot`

[3] They found TreeTagger was closely followed by HunPos, a re-implementation of TnT, and Stanford and ClearNLP were lower ranked. In an initial investigation, we compared Tnt, HunPos and TreeTagger and found Tnt to be consistently better than Treetagger, Hunpos followed closely but crashed on some languages (e.g., Arabic).

	BASELINES		BI-LSTM using:				$\vec{w}+\vec{c}$ +POLYGLOT		OOV ACC		BTS
	TNT	CRF	$\vec{w}$	$\vec{c}$	$\vec{c}+\vec{b}$	$\vec{w}+\vec{c}$	bi-LSTM	FREQBIN	bi-LSTM	FREQBIN	
avg	94.61	94.27	92.37	94.29	94.01	96.08†	**96.50**	**96.50**	87.80	87.98	95.70
Indoeur.	94.70	94.58	92.72	94.58	94.28	96.24†	**96.63**	96.61	87.47	87.63	–
non-Indo.	94.57	93.62	91.97	93.51	93.16	95.70†	96.21	**96.28**	90.26	90.39	–
Germanic	93.27	93.21	91.18	92.89	92.59	94.97†	**95.55**	95.49	85.58	85.45	–
Romance	95.37	95.53	94.71	94.76	94.49	95.63†	**96.93**	**96.93**	85.84	86.07	–
Slavic	95.64	94.96	91.79	96.45	96.26	97.23†	97.42	**97.43**	91.48	91.69	–
ar	97.82	97.56	95.48	98.68	98.43	98.89	98.87	**98.91**	95.90	96.21	–
bg	96.84	96.36	95.12	97.89	97.78	**98.25**	98.23	90.06	90.06	90.56	97.84
cs	96.82	96.56	93.77	96.38	96.08	97.93	**98.02**	97.89	91.65	91.30	98.50
da	94.29	93.83	91.96	95.12	94.88	95.94	96.16	**96.35**	86.13	86.35	95.52
de	92.64	91.38	90.33	90.02	90.11	93.11	**93.51**	93.38	85.37	86.77	92.87
en	92.66	93.35	92.10	91.62	91.57	94.61	**95.17**	95.16	80.28	80.11	93.87
es	94.55	94.23	93.60	93.06	92.29	95.34	95.67	**95.74**	79.26	79.27	95.80
eu	93.35	91.63	88.00	92.48	92.72	94.91	95.38	**95.51**	83.55	84.30	–
fa	95.98	95.65	95.31	95.82	95.03	96.89	**97.60**	97.49	88.82	89.05	96.82
fi	93.59	90.32	87.95	90.25	89.15	95.18	95.74	**95.85**	88.35	88.85	95.48
fr	94.51	95.14	94.44	94.39	93.69	96.04	**96.20**	96.11	82.79	83.54	95.75
he	93.71	93.63	93.97	93.74	93.58	95.92	96.92	**96.96**	88.75	88.83	–
hi	94.53	96.00	95.99	93.40	92.99	96.64	96.97	**97.10**	83.98	85.27	–
hr	94.06	93.16	89.24	95.32	94.47	95.59	96.27	**96.82**	90.50	92.71	–
id	93.16	92.96	90.48	91.37	91.46	92.79	93.32	**93.41**	88.03	87.67	92.85
it	96.16	96.43	96.57	95.62	95.77	97.64	97.90	**97.95**	89.15	89.15	97.56
nl	88.54	90.03	84.96	89.11	87.74	92.07	92.82	**93.30**	78.61	75.95	–
no	96.31	96.21	94.39	95.87	95.75	97.77	**98.06**	98.03	93.56	93.75	–
pl	95.57	93.96	89.73	95.80	96.19	96.62	**97.63**	97.62	95.00	94.94	–
pt	96.27	96.32	94.24	95.96	96.2	97.48	**97.94**	97.90	92.16	92.33	–
sl	94.92	94.77	91.09	96.87	96.77	97.78	**96.97**	96.84	90.19	88.94	–
sv	95.19	94.45	93.32	95.57	95.5	96.30	96.60	**96.69**	89.53	89.80	95.57

Table 2: Tagging accuracies on UD 1.2 test sets. $\vec{w}$: words, $\vec{c}$: characters, $\vec{b}$: bytes. Bold/†: best accuracy/representation; +POLYGLOT: using pre-trained embeddings. FREQBIN: our multi-task model. OOV ACC: accuracies on OOVs. BTS: best results in Gillick et al. (2016) (not strictly comparable).

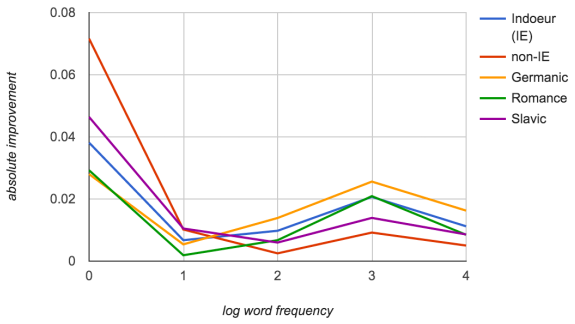

Figure 2: Absolute improvements of bi-LSTM ($\vec{w}+\vec{c}$) over TNT vs mean log frequency.

WSJ	Accuracy
Convnet (Santos and Zadrozny, 2014)	97.32
Convnet reimplementation (Ling et al., 2015)	96.80
Bi-RNN (Ling et al., 2015)	95.93
Bi-LSTM (Ling et al., 2015)	97.36
Our bi-LSTM $_{\vec{w}+\vec{c}}$	97.22

Table 3: Comparison POS accuracy on WSJ; bi-LSTM: 30 epochs, σ=0.3, no POLYGLOT.

Data set size Prior work mostly used large data sets when applying neural network based approaches (Zhang et al., 2015). We evaluate how brittle such models are with respect to their more traditional counterparts by training bi-LSTM ($\vec{w}+\vec{c}$ without Polyglot embeddings) for increasing amounts of training instances (number of sentences). The learning curves in Figure 3 show similar trends across language families.[4] TNT is better with little data, bi-LSTM is better with more data, and bi-LSTM always wins over CRF. The bi-LSTM model performs already surprisingly well after only 500 training sentences. For non-Indoeuropean languages it is on par and above

[4]We observe the same pattern with more, 40, iterations.

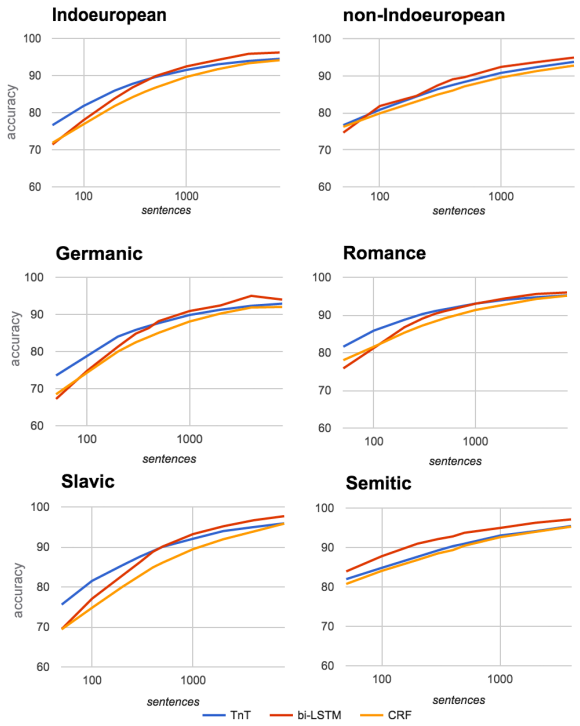

Figure 3: Amount of training data (number of sentences) vs tagging accuracy.

the other taggers with even less data (100 sentences). This shows that the bi-LSTMs often needs more data than the generative markovian model, but this is definitely less than what we expected.

Label Noise We investigated the susceptibility of the models to noise, by artificially corrupting training labels. Our initial results show that at low noise rates, bi-LSTMs and TnT are affected similarly, their accuracies drop to a similar degree. Only at higher noise levels (more than 30% corrupted labels), bi-LSTMs are less robust, showing higher drops in accuracy compared to TnT. This is the case for all investigated language families.

4 Related Work

Character embeddings were first introduced by Sutskever et al. (2011) for language modeling. Early applications include text classification (Chrupała, 2013; Zhang et al., 2015). Recently, these representations were successfully applied to a range of structured prediction tasks. For POS tagging, Santos and Zadrozny (2014) were the first to propose character-based models. They use a convolutional neural network (CNN; or convnet) and evaluated their model on English (PTB) and Portuguese, showing that the model achieves state-of-the-art performance close to taggers using carefully designed feature templates. Ling et al. (2015) extend this line and compare a novel bi-LSTM model, learning word representations through character embeddings. They evaluate their model on a language modeling and POS tagging setup, and show that bi-LSTMs outperform the CNN approach of Santos and Zadrozny (2014). Similarly, Labeau et al. (2015) evaluate character embeddings for German. Bi-LSTMs for POS tagging are also reported in Wang et al. (2015), however, they only explore word embeddings, orthographic information and evaluate on WSJ only. A related study is Cheng et al. (2015) who propose a multi-task RNN for named entity recognition by jointly predicting the next token and current token's name label. Our model is simpler, it uses a very coarse set of labels rather then integrating an entire language modeling task which is computationally more expensive. An interesting recent study is Gillick et al. (2016), they build a single byte-to-span model for multiple languages based on a sequence-to-sequence RNN (Sutskever et al., 2014) achieving impressive results. We would like to extend this work in their direction.

5 Conclusions

We evaluated token and subtoken-level representations for neural network-based part-of-speech tagging across 22 languages and proposed a novel multi-task bi-LSTM with auxiliary loss. The auxiliary loss is effective at improving the accuracy of rare words.

Subtoken representations are necessary to obtain a state-of-the-art POS tagger, and character embeddings are particularly helpful for non-Indoeuropean and Slavic languages.

Combining them with word embeddings in a hierarchical network provides the best representation. The bi-LSTM tagger is as effective as the CRF and HMM taggers with already as little as 500 training sentences, but is less robust to label noise (at higher noise rates).

Acknowledgments

We thank the anonymous reviewers for their feedback. AS is funded by the ERC Starting Grant LOWLANDS No. 313695. YG is supported by The Israeli Science Foundation (grant number 1555/15) and a Google Research Award.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed Word Representations for Multilingual NLP. In *CoNLL*.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved Transition-based Parsing by Modeling Characters instead of Words with LSTMs. In *EMNLP*.

Thorsten Brants. 2000. Tnt: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2015. Open-Domain Name Error Detection using a Multi-Task RNN. In *EMNLP*.

Kyunghyun Cho. 2015. Natural language understanding with distributed representation. *ArXiv*, abs/1511.07916.

Grzegorz Chrupała. 2013. Text segmentation with character-level text embeddings. In *Workshop on Deep Learning for Audio, Speech and Language Processing, ICML*.

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models. In *EMNLP*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *NAACL*.

Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *ArXiv*, abs/1510.00726.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Tobias Horsmann, Nicolai Erbs, and Torsten Zesch. 2015. Fast or accurate?–a comparative evaluation of pos tagging models. In *Proceedings of the International Conference of the German Society for Computational Linguistics and Language Technology*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

E. Kiperwasser and Y. Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *ArXiv e-prints*.

Matthieu Labeau, Kevin Löser, and Alexandre Allauzen. 2015. Non-lexical neural architecture for fine-grained pos tagging. In *EMNLP*.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *EMNLP*.

Christopher D Manning. 2011. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing*. Springer.

Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, Sam Bowman, Giuseppe G. A. Celano, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Berta Gonzales, Bruno Guillaume, Jan Hajič, Dag Haug, Radu Ion, Elena Irimia, Anders Johannsen, Hiroshi Kanayama, Jenna Kanerva, Simon Krek, Veronika Laippala, Alessandro Lenci, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Ctlina Mrnduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Shunsuke Mori, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Prokopis Prokopidis, Sampo Pyysalo, Loganathan Ramasamy, Rudolf Rosa, Shadi Saleh, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Jan Štěpánek, Alane Suhr, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Sumire Uematsu, Larraitz Uria, Viktor Varga, Veronika Vincze, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2015. Universal dependencies 1.2. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Barbara Plank, Dirk Hovy, Ryan McDonald, and Anders Søgaard. 2014. Adapting taggers to twitter using not-so-distant supervision. In *COLING*.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *ICML*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *pre-print*, abs/1510.06168.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *ACL*.