MULTI-TASK LEARNING FOR NEURAL DEPENDENCY PARSING

Laura Ruis
School of Informatics
University of Amsterdam
laura.ruis@student.uva.nl

ABSTRACT

In this work we investigate the effect of integrating a POS tagger with a neural dependency parser, taking a multi-task learning approach to parsing. This approach has the benefit that after training, POS tags are no longer needed. The architecture is based on the graph-based deep biaffine parser proposed by Dozat & Manning (2016). Instead of training a separate POS tagger, the assigning of POS tags and parsing is done by the same model. This should improve generalization. We perform experiments on the Universal Dependencies datasets for English and Mandarin Chinese, comparing several models that use POS tags as input to models that do not need this, but instead take a multi-task learning approach. We find empirical evidence suggesting that the latter approach is promising, but needs further work in terms of POS tag accuracy, which is higher for the model that trains a separate POS tagger and uses the predicted tags as input. We suspect that if the multi-task learning model can predict tags with the same accuracy as the separate POS tagger, it can also reach similar or even better performance than the model that uses tags as input, depending on the language that is parsed.

1 INTRODUCTION

Dependency parsing is the task of finding and labeling relations among words in a sentence. Graph-based parsers treat this task as the problem of learning a score function over dependency trees such that the correct tree is given the highest score. In this work we integrate a neural graph-based dependency parser with a part-of-speech tagger. The architecture of the parser is based on the deep biaffine parser (Dozat & Manning, 2016) and its extension used in the CoNLL shared task of 2017 (Dozat et al., 2017), where the authors add a character-based model to deal with languages with rich morphology (described in Sections 2.1 - 2.3 of this paper). Dozat et al. showed that POS tags are crucial for parsing performance and for the CoNLL shared task[1] they train a separate POS tagger and use the predicted tags as input to the model. The contributions we make in this work are twofold; we take a multi-task learning approach and predict POS tags from the first layer of the bi-LSTM used in the encoder (Section 2.4), we then evaluate the models on English and Mandarin Chinese (Sections 3.1 - 3.2). Multi-task learning has been shown to improve generalization performance in many domains (Caruana, 1997) as well as dependency parsing specifically (Hashimoto et al., 2016). Integrating the POS tagger with the parser moreover has the benefit that POS tags are no longer needed after training, making it an arguably simpler approach.

2 ARCHITECTURE

2.1 BACKGROUND

While traditional graph-based parsers made use of hand-crafted features, a successful switch to learning dense features was made by Chen & Manning (2014), who incorporated deep neural networks in their parsers. The need for hand-crafting features was further reduced by Kiperwasser & Goldberg (2016). Their dependency parsers simplify the problem of finding good features to learning them along with a parser. The architecture involves a bidirectional long short-term memory

[1] http://universaldependencies.org/conll17/

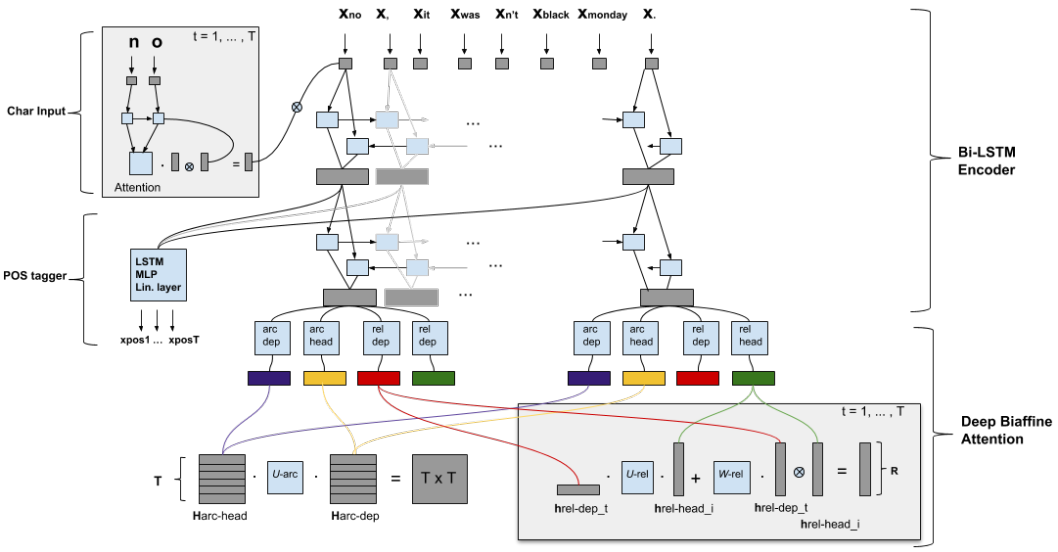

Figure 1: Full model architecture (input of ROOT is left out for simplicity).

recurrent neural network (Hochreiter & Schmidhuber, 1997; Schuster & Paliwal, 1997) to extract feature representations that are subsequently scored with a multi-layer perceptron attention mechanism and used for parsing. Dozat & Manning (2016) modified the MLP attention mechanism by using something they call deep biaffine attention.

2.2 Deep Biaffine Parser

The architecture of the deep biaffine parser proposed by Dozat & Manning is similar to the graph-based dependency parser by Kiperwasser & Goldberg in the sense that it also uses a bi-LSTM encoder as a feature extractor that is trained along with the parser. The difference lies in the method of mapping the outputs of the encoder to vectors that can be used for parsing. The calculations done by the encoder can be captured for a single word (time step) by the following equations, where the notation is taken to be similar to Dozat et al. (2017) for clarity.

$$\mathbf{x}_t = \mathbf{e}(w_t) \oplus \mathbf{e}(p_t) \tag{1}$$

$$\mathbf{r}_t = \text{LSTM}_{fwd}(\mathbf{x}_{0:T})_t \oplus \text{LSTM}_{bwd}(\mathbf{x}_{0:T})_t \tag{2}$$

$$\mathbf{h}_t^{(\text{i})} = \text{MLP}^{(\text{i})}(\mathbf{r}_t) \qquad \text{for } i = \text{arc-dep, arc-head, rel-dep, rel-head} \tag{3}$$

The words and POS tags of a sentence are embedded in vector representations ($\mathbf{e}(w_t)$ and $\mathbf{e}(p_t)$, resp.), concatenated and passed through the bi-LSTM. The output at each time step is a concatenation of its forward and backward state. Dozat & Manning note that the feature representations extracted by the bi-LSTM network might contain more information than necessary for the task at hand, and could benefit from dimensionality reduction. To this end the states at each time step are passed through four different MLP's to produce four different representations before they are used in the classifier; hence the term 'deep attention'. There are two tasks that need to be addressed by these vectors. First, a word needs to find its syntactic head. The classifier uses $\mathbf{h}_{0:T}^{arc-dep}$ and $\mathbf{h}_{0:T}^{arc-head}$ to this end, as shown in equation 4. Second, the relation between a word and its head needs to be labeled, for which the classifier uses $\mathbf{h}_{0:T}^{rel-dep}$ and $\mathbf{h}_{0:T}^{rel-head}$, shown in equation 5.

$$\mathbf{s}_t^{(\text{arc})} = H^{(\text{arc-head})} U^{(\text{arc})} \mathbf{h}_t^{(\text{arc-dep})} + H^{(\text{arc-head})} \mathbf{u} \tag{4}$$

$$\mathbf{s}_t^{(\text{label})} = \mathbf{h}_i^{T(\text{rel-head})} U^{(\text{rel})} \mathbf{h}_t^{(\text{rel-dep})} + W^{(\text{rel})} (\mathbf{h}_t^{(\text{rel-dep})} \oplus \mathbf{h}_i^{(\text{rel-head})}) + \mathbf{b}^{(\text{rel})} \tag{5}$$

$U^{(\text{arc})}$, $U^{(\text{rel})}$ and $W^{(\text{rel})}$ are matrices of parameters that need to be learned and $H^{(\text{arc-head})}$ is a $(T+1) \times D$ matrix containing the representations of all words in the sentence, including the root. D represents the output dimension of the MLP's. A head word w_i for every modifier word w_t is found by taking the argument $i \in 0, \ldots, T$ that maximizes $\mathbf{s}_{t,i}^{(\text{arc})}$, with 0 the root. A label for each relation in the sentence is then found by taking the argument $j \in 1, \ldots, R$ that maximizes $\mathbf{s}_{t,j}^{(\text{rel})}$, where R is the number of possible labels. In Figure 1 the equations are shown graphically.

Hyper-parameter	Value	Hyper-parameter	Value
Word, POS, char embeds	100	Optimizer	Adam
Bi-LSTM layers, size	3, 1024	Initial LR	0.001
(Bi-)LSTM chars layers, size	1, 200	β_1, β_2	0.9
LSTM tagger layers, size	1, 128	Decay rate	0.75
POS embeds (MTL)	28	Gradient clipping	5.0
MLP's arc, label size	512, 128	Batch size	2000 words
Dropout	0.33	Iterations	60.000

Table 1: Hyper-parameter settings.

2.3 Character Model

Languages where syntactic relations between words are indicated through inflection might benefit from more than just word-level representations. It has been shown that in order to parse languages with rich morphology, it is useful to add a model that extracts character-level representations. These representations can be extracted in many ways (Dozat et al., 2017; Peters et al., 2018; Ma et al., 2018). Dozat et al. build word representations from character embeddings that are encoded by an LSTM and converted into a fixed length vector with an attention mechanism over the produced hidden states and the final cell state. The adjustment we make to the described character model is instead of using an element-wise sum of word- and character-level embeddings as input to the parser, the representations are concatenated and projected back to the original dimension of the word embeddings. This should give the model the opportunity to ignore certain representations if they are not useful. In principle, the final cell state of the LSTM that encodes the characters of a word should contain sufficient information about all the characters. Therefore, we also implement a simplification of the character model, where instead of the attention mechanism the final cell states of a bi-LSTM are concatenated and linearly transformed to the desired dimensionality.

2.4 Multi-Task Learning

Multi-task learning is an inductive transfer technique focused on leveraging knowledge learned by one task for another task. The underlying motivation for having the same model learn several tasks in parallel is that the shared representation is a form of regularization. Learning several tasks with a model that shares representations has been shown to improve generalization in various domains (Caruana, 1997) as well as for dependency parsing specifically (Hashimoto et al., 2016). Dozat & Manning train a separate POS tagger to provide the text with POS tags prior to parsing it, and use the predicted tags as input to the model. We take a multi-task learning approach, where instead of using predicted POS tags as input to the model, feature representations of the words are used to train a POS tagger along with the parser. It has been shown to be effective to use low-level representations for low-level tasks like POS tagging (Søgaard & Goldberg, 2016; Peters et al., 2018), therefore we extract feature representations from the first layer of the bi-LSTM encoder. These representations are then passed through an LSTM decoder, where each predicted tag is conditioned on the previously predicted tags. The assigning of a POS tag to each word is done with the same affine classification as used in Dozat et al. (2017), namely a MLP followed by a linear layer.

3 Experiments

3.1 Experimental Setup

The models are trained on the Universal Dependencies datasets for English and Mandarin Chinese (Nivre et al., 2018) that are used in the CoNLL Shared Task of 2018[2] and segmentation is done by the UDPipe (Straka et al., 2016). Models that use POS tags as input use predicted tags (obtained from Dozat & Manning[3]). Models trained with MTL use gold POS tags for back propagation. The parameters are optimized with Adam (Kingma & Ba, 2014) with a β_1 and β_2 of 0.9, empirically found to improve performance over the settings that Kingma & Ba recommend by Dozat & Manning.

[2]http://universaldependencies.org/conll18/

[3]https://nlp.stanford.edu/data/CoNLL17_Parser_Models/

The learning rate decay scheme is taken from Ma et al. (2018), meaning the learning rate is annealed with a fixed decay each time the F1 score of UAS and LAS stops increasing on the validation set. The tagger is trained with teacher-forcing (Williams & Zipser, 1989) and at test time the LSTM uses predicted tags that are concatenated with the output of the first layer of the bi-LSTM of the parser as input. Since POS tagging is a substantially easier task than parsing and converges a lot faster, different weights for the tag loss are used during training, namely $\alpha \in \{1, 0.1, 0.01\}$. We initialize the word embeddings with GloVe vectors (Pennington et al., 2014). We apply dropout throughout the model, as well as on the recurrent connections of the bi-LSTM's (Gal & Ghahramani, 2016) and we drop entire words and POS tags during training as described by Dozat & Manning. In the character model a (recurrent) dropout of 0.33 is applied to the (bi-)LSTM. In the POS tagger regular dropout is applied but recurrent dropout is not. A summary of the parameter settings are shown in Table 1.

3.2 RESULTS

In Table 2 the results of the experiments on the development set can be seen. The table shows the accuracy of the POS tags ('POS'), the the percentage of words that are assigned the correct syntactic head (unlabeled attachment score, 'UAS'), and the percentage of words that are assigned both the correct syntactic head and corresponding dependency label (labeled attachment score, 'LAS'). The baseline model is the UDPipe, all the other rows are results by the reimplementation of the deep biaffine parser and its extensions. The model with the best performance on English text is the simplest form of the reimplementation as described in Section 2.2 and for Chinese text the model that uses the attention character model as described in Section 2.3. Adding character representations seems to hurt performance for English and boost it for Chinese. This result is surprising, since the Chinese language contains no morphological inflection, whereas English does. Comparing the simple character model to the one with attention we observe that for English the simpler character model performs better, and for Chinese the one with attention performs better. This result could be explained by the vocabulary size. English has a very small number of characters in this dataset, namely 113, whereas in the Chinese dataset the number of characters is much larger, namely 3575. This might mean that a model trying to parse a language with a large vocabulary size, like Chinese, benefits from a more expressive character model.

The models that use multi-task learning instead of POS tag input have a lower POS tag accuracy, UAS and LAS. Lower weights for the POS tag loss (α) only make this worse. To be able to compare the models that use MTL as opposed to POS tag input we provide the training and development set with predicted POS tags obtained by the best MTL model (with $\alpha = 1$) and train the D&M reimplementation on this data (results are shown in the row named 'D&M reimp. + tag input from MTL'). We now observe an UAS and LAS that are much closer to the models that use MTL. These results support the empirically found evidence by Dozat et al. that POS tag accuracy is very important for parsing performance. For English, the model that uses tags as input still outperforms the model that uses MTL, but for Chinese we observe the opposite. An explanation for this might be that POS tags are especially important for Chinese parsing, an observation also made by Ma & Hovy (2017). This makes intuitive sense: Chinese characters contain no morphological inflection and a POS tag could be crucial extra information for finding dependency structures.

In Table 3 the results on the test set are shown. Again, all results below the double vertical line are obtained by our own implemented models. The row named 'D&M reimp. + best chars' refers to the model with the attention character model for Chinese, and the model with the simple character model for English. All models significantly outperform the baseline, but do not outperform the results reported in Dozat et al. (2017).

4 CONCLUSION

In this work we describe and implement a simple graph-based dependency parser and evaluate several variations of it on the Universal Dependencies datasets for English and Mandarin Chinese. The basis of the architecture is the deep biaffine parser (Dozat et al., 2017). The goal of the paper is to evaluate the performance of the end-to-end parser that takes a multi-task learning approach to parsing and predicts POS tags with the same model, meaning it does not take POS tags as input. All models outperform the baseline model, namely the UDPipe. The MTL models without POS tag

Language	*English (en-wst)*			*Mandarin (zh)*		
Model	**POS**	**UAS**	**LAS**	**POS**	**UAS**	**LAS**
Baseline UDPipe	92.90	80.80	77.60	83.20	61.10	57.40
D&M reimplementation	96.15*	**90.27**	**87.64**	94.78*	82.69	78.33
D&M reimp. + attention character model	96.15*	89.96	87.26	94.78*	**83.60**	**79.22**
D&M reimp. + simple char. model	96.15*	90.28	87.55	94.78*	83.41	79.11
D&M reimp. + tag input from MTL	94.32	89.22	86.08	91.15	79.74	74.30
D&M reimp. + attn. char. + MTL α 1	94.32	88.94	85.86	91.15	81.09	76.00
D&M reimp. + attn. char. + MTL α 0.1	93.78	88.93	85.66	89.71	80.99	75.87
D&M reimp. + attn. char. + MTL α 0.01	92.74	88.69	85.50	87.32	80.29	74.97

Table 2: Results on Development sets for English and Mandarin Chinese (*POS tag accuracy by Dozat & Manning).

Language	*English (en-wst)*			*Mandarin (zh)*		
Model	**POS**	**UAS**	**LAS**	**POS**	**UAS**	**LAS**
Baseline UDPipe*	92.42	78.87	75.84	82.59	61.50	57.40
Dozat & Manning*	94.82	**84.74**	**82.23**	85.07	**68.95**	**65.88**
D&M reimp.	94.82	84.48	81.98	85.07	67.98	64.92
D&M reimp. + best chars	94.82	84.17	81.69	85.07	68.13	65.07
D&M reimp + attn. char. + MTL α 1	92.98	83.21	80.47	82.28	66.28	62.73

Table 3: Results on test sets for English and Mandarin Chinese (*taken from `http://universaldependencies.org/conll17/results.html`).

input do not manage to outperform the result mentioned in Dozat et al. (2017). We suspect that the reason for this is the fact that the POS tags used as input to the models have a lower error rate than the tags predicted by the models that use multi-task learning. This suspicion is based on a model that is trained with POS tags as input that have a similar error rate as the POS tags predicted by the MTL model. When comparing the best MTL model without POS tag input to this model, we observe much closer performance. The MTL model still performs slightly worse for English, but performs better for Chinese. This suggests that languages for which POS tags are very important for parsing, like Chinese, can benefit from taking a multi-task learning approach to parsing, as opposed to training a separate POS tagger. This claim needs more evidence that can be provided in future work where the POS tagger in the multi-task learning model is improved upon.

For English parsing we do not observe an improvement in performance when adding character models, which contradicts results found by Dozat et al. (2017). It might be worth to improve the architecture that extracts character representations, for example with the CNN-based model described by Ma et al. (2018), which they finds to boost performance of the deep biaffine parser on all datasets.

REFERENCES

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, Jul 1997. ISSN 1573-0565. doi: 10.1023/A:1007379606734. URL `https://doi.org/10.1023/A:1007379606734`.

Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 740–750. Association for Computational Linguistics, 2014. doi: 10.3115/v1/D14-1082. URL `http://www.aclweb.org/anthology/D14-1082`.

Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734, 2016. URL `http://arxiv.org/abs/1611.01734`.

Timothy Dozat, Peng Qi, and Christoper D. Manning. Stanford's graph-based neural dependency parser at the conll 2017 shared task. pp. 20–30, 01 2017.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*, pp. 1019–1027, 2016.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple NLP tasks. *CoRR*, abs/1611.01587, 2016. URL http://arxiv.org/abs/1611.01587.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *CoRR*, abs/1603.04351, 2016. URL http://arxiv.org/abs/1603.04351.

Xuezhe Ma and Eduard H. Hovy. Neural probabilistic model for non-projective MST parsing. *CoRR*, abs/1701.00874, 2017. URL http://arxiv.org/abs/1701.00874.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. Stack-pointer networks for dependency parsing. In *The 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia, July 2018. URL https://arxiv.org/abs/1805.01087.

Joakim Nivre et al. Universal Dependencies 2.2, 2018. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague, http://hdl.handle.net/11234/SUPPLYTHENEWPERMANENTIDHERE!

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November 1997. ISSN 1053-587X. doi: 10.1109/78.650093. URL http://dx.doi.org/10.1109/78.650093.

Anders Søgaard and Yoav Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. pp. 231–235, 01 2016.

Milan Straka, Jan Hajič, and Jana Straková. UDPipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, Portoro, Slovenia, 2016. European Language Resources Association. ISBN 978-2-9517408-9-1.

R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.