Bootstrapping a neural net dependency parser for German using CLARIN resources

Daniël de Kok
University of Tübingen
Wilhelmstraße 19
72074 Tübingen
daniel.de-kok@uni-tuebingen.de

Abstract

Statistical dependency parsers have quickly gained popularity in the last decade by providing a good trade-off between parsing accuracy and parsing speed. Such parsers usually rely on hand-crafted symbolic features and linear discriminative classifiers to make attachment choices. Recent work replaces these with dense word embeddings and neural nets with great success for parsing English and Chinese. In the present work, we report on our experiences with neural net dependency parsing for German using CLARIN data.

1 Introduction

A dependency parser transforms a sentence w_0^n with the artificial root token w_0 in a dependency graph $G =< V, A >, V \subseteq w_0^n$ and $A \subseteq V \times R \times V$, wherein R is a set of dependency labels. A transitional dependency parser (Nivre, 2003; Kübler et al., 2009) performs this transduction using a transition system, where the states are configurations consisting of a buffer β of unprocessed tokens, a stack σ of partially processed tokens, and A the set of dependency arcs that were found so far. The transitions transform the configuration, typically by introducing a leftward arc, a rightward arc, or moving a token from β to σ. Transformations are applied until the system reaches a final configuration wherein β is empty and σ only contains the artificial root token.

In a particular configuration c, there are often multiple possible transitions. A classifier $\lambda(t|c)$ is used to find the most probable transition t in a configuration c. If the parser only considers the most probable transition, parsing time is linear in the commonly-used transition systems. The classification function λ is typically a linear classifier that is parametrized using features that decompose the configuration that is under consideration. λ is trained by running a parser over the training data using an oracle that produces a sequence of transitions that results in $A = A_{gold}$.

2 Neural net parsers

Chen and Manning (2014) propose a transition-based parser that uses a neural net for λ. Moreover, the input of the parser does not consist of symbolic features, but the concatenation of (dense) word, tag, and relation embeddings of tokens that are in interesting positions of the parser configuration.

Their approach has three potential benefits over dependency parsing with a linear model: (1) the use of a non-linear activation function in the hidden layer(s) avoids the need to carefully craft effective feature combinations; (2) the use of word embeddings can counter data sparseness, e.g. *dog* and *puppy* will behave similarly with regards to syntax and selectional preferences; and (3) transition-based dependency parsers that use symbolic features are known to spend most of their time on feature extraction (Bohnet, 2010), while the use of embeddings only requires a small number of lookups and fast dense matrix-vector multiplications.

Chen and Manning (2014) report large improvements in accuracy over existing transition-based parsers such as Malt and the graph-based MSTParser for English and Chinese. Of course, their work leaves some interesting questions: how well does this approach work for other languages and how stable are the network topology and chosen hyperparameters? Our goal is to reproduce this approach for German using CLARIN resources, with the twofold goal of validating their results and bootstrapping a neural net dependency parser for German.

3 Training of embeddings

Since the input of a neural net parser consists solely of embeddings, it is of prime importance that the embeddings are of a high quality. It is well-known that such embeddings can only be obtained using huge training sets (Mikolov et al., 2013).

Although a vast amount of textual data for German is available through the CLARIN-D project,[1] we also require part-of-speech and morphological annotations without any divergence in annotation schemes. To this end, we use two large corpora for German: TüBa-D/W (de Kok, 2014) and TüPP-D/Z (Müller, 2004). The TüBa-D/W is a 615 million token treebank of German Wikipedia text that provides part-of-speech and morphological annotations. The TüPP-D/Z is a 204 million token corpus of text from the German newspaper Taz. To ensure that the annotation schemes for part-of-speech tags and morphology are the same as in TüBa-D/W, we apply the annotation pipeline described in De Kok (2014), except that we retain the token and sentence boundaries of TüPP-D/Z. We also remove sentences that are part of TüBa-D/Z (Telljohann et al., 2004). Our training data for the embeddings consist of the concatenation of these two corpora (818 million tokens, 48 million sentences). We prepend a special root token to each sentence that is used internally by the parser.

We train the embeddings using an adaptation of Word2Vec (Mikolov et al., 2013) that uses a structured variant of the skip-gram model (Ling et al., 2015). This adaptation uses a dedicated output matrix for each position in token's context window. As a result, the embeddings capture order dependence, making them better suited for syntax-based tasks. We use the same training hyperparameters as proposed in Ling et al. (2015) and a minimum token frequency of 30.[2] Experiments with other hyperparameters did not provide a consistent improvement.

The dependency parsers were trained and evaluated using the dependency version (Versley, 2005) of the TüBa-D/Z (Telljohann et al., 2004). We first split the treebank into five (interleaved) parts. Two parts were used for training and validation. The remaining three parts were held out and used for the evaluation. The morphology annotations in TüBa-D/Z are replaced by annotations provided by RFTagger (Schmid and Laws, 2008) to use the same tag set as TüBa-D/W and the reprocessed TüPP-D/Z.

4 Parser configurations

4.1 SVM parser

We use the dpar[3] parser as the 'traditional' parser in our comparison. dpar is a linear SVM-based parser that is architecturally similar to MaltParser (Nivre et al., 2006), adding hash kernels (Shi et al., 2009; Bohnet, 2010), and providing more fine-grained control over morphology features. We use this parser and the neural net parser with the stack-projective transition system.

The initial set of feature templates is obtained by applying MaltOptimizer (Ballesteros and Nivre, 2012) to the training and validation data. We then add extra hand-crafted morphology-based feature templates,[4] such as: take the *number* morphological feature of σ_1 and σ_2 (the two tokens on top of the stack), and the *case* feature of token σ_1.

4.2 Neural net parser

Parser input The dparnn parser is used as the neural net parser in our comparison. Since a neural net that uses a non-linear activation in its hidden layer can infer complex features, the parser does not need feature templates. Instead, we specify the positions of the stack (σ_0^{n-1}) and buffer (β_0^{n-1}) that the parser is allowed to use, together with the information layers. The possible layers are: TOKEN, TAG, MORPH, and DEPREL (the

[1] `http://clarin-d.de/de/auffinden/referenzressourcen`

[2] The embeddings are available at: `http://hdl.handle.net/11022/0000-0000-8507-2`

[3] `http://github.com/danieldk/dpar`

[4] The feature template specification is included in the *dpar* repository.

relation to a token's head). In addition, the LDEP and RDEP indirections can be used, which give the leftmost and rightmost dependent of a token. For feature formation, we provide the parser with the following: TOKEN(σ_0^3), TOKEN(β_0^2), TAG(σ_0^3), TAG(β_0^2), TOKEN(LDEP(σ_0^1)), TOKEN(RDEP(σ_0^1)), TAG(LDEP(σ_0^1)), TAG(RDEP(σ_0^1)), DEPREL(σ_0), DEPREL(LDEP(σ_0^1)), and DEPREL(RDEP(σ_0^1)). For experiments with morphology, we also give the parser access to MORPH(σ_0^1) and MORPH(β_0).

Embeddings For word embeddings and morphology embeddings, we always use the embeddings obtained through unsupervised training as described in Section 3. The embeddings for dependency relations are trained during the training of the parser, by learning weights through backpropagation to the relation vectors. For part-of-speech tags, we explore two possibilities: learning embeddings unsupervised from a large corpus following Section 3 and learning embeddings through backpropagation while training the parser. Chen and Manning (2014) did not explore unsupervised training of tag embeddings nor unsupervised training of morphology embeddings.

Training The training data for the neural network is obtained by running the parser over the training data using an oracle which extracts each transition and the corresponding parser input. We normalize each embedding using its ℓ_2 norm and the input vector to the neural net per index by its mean and variance.

We use Caffe (Jia et al., 2014) to train the network. Since Caffe was developed for image processing, we represent each input as an 1×1 image with the number of channels that is equal to the input size. To train tag and relation embeddings, we take the following approach: tags and dependency relations are encoded using one-hot encoding in the input. We create a hidden linear layer for each input position of the form TAG$(\cdot)$ and DEPREL$(\cdot)$. The weights of layers of a particular type (TAG and DEPREL) are shared. The output of these hidden layers, along with the remainder of the input, is the input of a hidden layer with a non-linear activation function. After training, the aforementioned hidden linear layers contain the tag and relation embeddings – the n-th weight of the m-th neuron stores the m-th element of the embedding vector of the tag/relation corresponding to the n-th one-hot bit. In our parser, we use 50 neurons in the embedding layers, to obtain embeddings of size 50. The network topology that is used during training is shown schematically in Figure 1. After training, we extract the embeddings, to simplify the network topology during parsing so that it does not need the hidden linear layers.

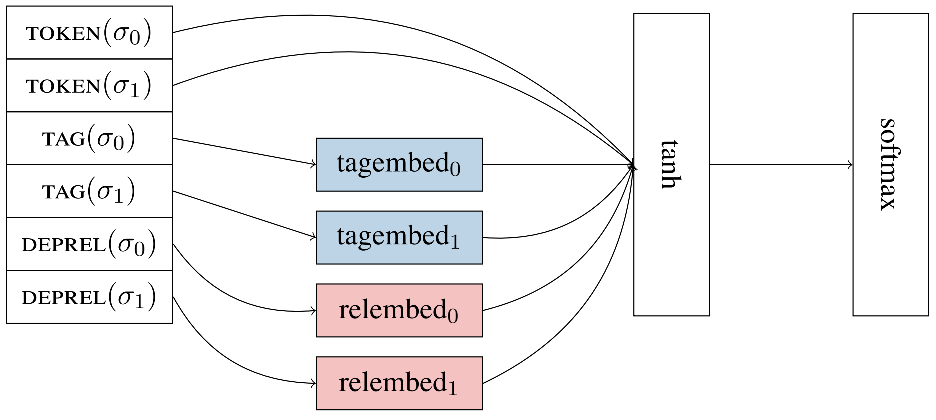

Figure 1: Network topology used during training, with supervised training of tag and relation embeddings. Embeddings with the same color have shared weights. The number of inputs is reduced for illustrative purposes.

As Chen and Manning (2014), we use a non-linear hidden layer with 200 neurons, an output layer that uses the softmax activation function, and the Adagrad solver (Duchi et al., 2011). However, their other hyperparameters, a cube activation function for the hidden layer and 50% dropout (Srivastava et al., 2014), did not work well in our experiments. Instead, opted for more conservative hyperparameters: a hyperbolic tangent activation function for the hidden layer, and mild input layer and hidden layer dropout of 10% and 5% respectively.

5 Results

In our discussion of the results, we will start with the most basic versions of both types of parsers: dpar (*dpar*) and dparnn trained with unsupervised word embeddings and supervised tag and relation embeddings (*dparnn STE*). Then, we will discuss dparnn with tag embeddings learned through unsupervised training (*dparnn UTE*). Finally, we will look at the effect of giving access to morphological information to both dpar (*dpar morph*) and dparnn (*dparnn UTE morph*).

As can be seen in Table 1, the *dparnn STE* parser clearly outperforms *dpar*. We expect that this is the case because the neural net is able to infer features that are difficult to get by hand-crafting feature templates or doing a greedy search such as that of MaltOptimizer. Moreover, since the word embeddings are trained on a very large corpus, the *dparnn STE* has much better lexical coverage (97.13% of tokens, 75.73% of types) of the evaluation data than *dpar* (89.32% of tokens, 30.67% of types).

Model	**LAS**
dpar	88.28
dpar (morphology)	88.85
dparnn STE	88.83
dparnn UTE	89.08
dparnn UTE + Morphology	**89.40**

Table 1: Results of parsing using SVM classifiers using symbolic features and neureal nets using word embeddings.

Using unsupervised tag embeddings improves the result by 0.25%. This is perhaps surprising, because the unsupervised tag embeddings were trained on non-gold standard data. However, it seems that the sheer amount of examples puts each tag more accurately in vector space.

Finally, adding morphological information to the mix improves the SVM parser profoundly (0.57%), while also providing an improvement for the neural net parser (0.32%). In the final parser configurations, the neural net parser outperforms the SVM parser by 0.55%.

6 Conclusion

In this work we have shown that neural net dependency parsers that rely on word embeddings outperform 'traditional' SVM dependency parsers on German. Since neural net parsers rely on large corpora of automatically annotated text, efforts to create and distribute such corpora are paramount to the improvement of the state of the art in parsing. In future work, we hope to address our poor man's treatment of morphology, by making morphological analysis an integral part of parsing.

References

[Ballesteros and Nivre2012] Miguel Ballesteros and Joakim Nivre. 2012. MaltOptimizer: an optimization tool for MaltParser. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 58–62. Association for Computational Linguistics.

[Bohnet2010] Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 89–97. Association for Computational Linguistics.

[Chen and Manning2014] Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 1, pages 740–750.

[de Kok2014] Daniël de Kok. 2014. TüBa-D/W: a large dependency treebank for german. In *Proceedings of the Thirteenth International Workshop on Treebanks and Linguistic Theories (TLT13)*, page 271.

[Duchi et al.2011] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

[Jia et al.2014] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*.

[Kübler et al.2009] Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127.

[Ling et al.2015] Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL), Denver, CO*.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

[Müller2004] Frank Henrik Müller. 2004. Stylebook for the tübingen partially parsed corpus of written German (TüPP-D/Z). In *Sonderforschungsbereich 441, Seminar für Sprachwissenschaft, Universität Tübingen*, volume 28, page 2006.

[Nivre et al.2006] Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, pages 2216–2219.

[Nivre2003] Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

[Schmid and Laws2008] 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*, pages 777–784. Association for Computational Linguistics.

[Shi et al.2009] Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, and SVN Vishwanathan. 2009. Hash kernels for structured data. *The Journal of Machine Learning Research*, 10:2615–2637.

[Srivastava et al.2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[Telljohann et al.2004] Heike Telljohann, Erhard Hinrichs, and Sandra Kübler. 2004. The TüBa-D/Z treebank: Annotating German with a context-free backbone. In *In Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004*. Citeseer.

[Versley2005] Yannick Versley. 2005. Parser evaluation across text types. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005)*.