Association metrics in neural transition-based dependency parsing

Patricia Fischer **Sebastian Pütz** **Daniël de Kok**
SFB 833
University of Tübingen, Germany
{patricia.fischer, sebastian.puetz, daniel.de-kok}@uni-tuebingen.de

Abstract

Lexical preferences encoded as association metrics have been shown to improve performance on structural ambiguities that are still challenging for modern parsers. This paper introduces a mechanism to include lexical preferences into a neural transition-based dependency parser for German. We compare pointwise mutual information (PMI) and embedding-based scores. Both the PMI-based model and the embedding-based model outperform the baseline significantly. The best model is PMI-based and increases overall performance by 0.26 LAS points over the baseline.

1 Introduction

Structural ambiguities that cannot be solved purely on the basis of structural preferences still pose a major challenge to syntactic parsing. Prepositional phrase (PP) attachment and subject-object inversion are two examples of such ambiguities. Table 1 gives an overview of the most frequent parser errors in a German newspaper corpus of 20K sentences and 350K tokens, parsed by the De Kok and Hinrichs (2016) parser with 92.01 labeled attachment score. It shows that more than one third of all errors involves prepositions, subjects and accusative objects.

Relation	**Error count**	**Percent of all errors**
Prepositional phrase/object	6,861	25.62
Adverbial	3,106	11.60
Conjunction	2,391	8.92
Accusative object	1,608	6.00
Subject	1,577	5.81
Total error count	26,775	100.00

Table 1: Five most frequent parser errors by dependency label of the parser by De Kok and Hinrichs (2016) for a German newspaper corpus. More than one third of all errors involves prepositions, subjects and accusative objects.

Resolving such ambiguities often requires context information or world knowledge. In Example 1, the direct object *Problem* 'problem' is fronted. The parser, however, learns from training data a preference for the unmarked word order with sentence-initial subject. *Problem* would therefore be misclassified as subject. Additionally, both *Problem* and *Post* 'post' are ambiguous between nominative and accusative case. Information on the sentence level thus does not suffice to decide on the correct attachment. Contextual knowledge reveals that *Problem* typically attaches to *lösen* 'to solve' as direct object.

Semantic preferences can provide further disambiguation cues. The verb *lösen* prefers an animate subject and an inanimate direct object. In Example 1, both *Problem* and *Post* are inanimate. World knowledge is necessary to interpret *Post* as the (animate) group of postal employees. Such knowledge can be learned from large corpora. Semantic preferences then yield the correct analysis of *Post* as animate subject and *Problem* as inanimate direct object of *lösen*.

(1) *Dieses Problem muß auch die Post noch lösen .*
This problem has-to also the post still solve .

'*The German Federal Post Office still has to solve this problem.*'

Pointwise mutual information (PMI, Fano (1961)) has been used to measure selectional preferences (Church and Hanks, 1990). PMI indicates how much two words occur together more often than chance. In the example above, a high PMI of *lösen* and *Problem* in $verb \rightarrow direct\ object$ relations would already provide enough information to solve the subject-object ambiguity. As PMIs are ideally calculated from large corpora, they provide additional context information.

In more traditional analyses of dependency distributions, it has been shown that PMI is very beneficial to solve structural ambiguities such as PP attachment (Hindle and Rooth, 1993; Ratnaparkhi, 1998; Volk, 2002). In parsing, bilexical preferences have been used by Van Noord (2007) to improve syntactic ambiguity resolution in a Maximum-Entropy parser for Dutch. Kiperwasser and Goldberg (2015) extended bilexical preferences to contextual association scores based on PMI and dependency embeddings (Levy and Goldberg, 2014a) in a graph-based parser. Mirroshandel and Nasr (2016) integrated selectional preferences into a graph-based dependency parser.

Recent approaches to neural dependency parsing (Chen and Manning, 2014; Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017) implicitly encode information about co-occurrences through vector representations of the token input (Mikolov et al., 2013). However, De Kok et al. (2017) have shown for PP attachment that neural models can still benefit from information provided by PMI scores.

This paper argues that bilexical preferences are also useful in neural transition-based dependency parsing. The two main contributions are 1) a methodology to apply bilexical preferences to neural transition-based dependency parsing, and 2) an evaluation of two types of association metrics in a neural dependency parser. Results confirm that association metrics benefit neural dependency parsing. The best association score models outperform the baseline by 0.26 LAS points and improve performance on two ambiguity solving tasks by up to 2.33 points.

2 Bilexical Preferences in Neural Dependency Parsing

2.1 Approach

Transition-based dependency parsing is the task of establishing dependency relations between tokens (Kübler et al., 2009). Typically, unprocessed tokens are put on a buffer β, and a stack σ keeps track of the partially processed tokens. In the transition system used in this work, sometimes called the stack-projective system, attachments are made between the token on top of the stack and the second token on the stack (Nivre, 2004). A LEFTARC transition attaches the second token on the stack as a dependent of the token on top of the stack with relation $r \in R$, and vice versa for a RIGHTARC transition:

$$\begin{aligned}
&\textsc{LeftArc} && (\sigma|i|j,\beta,R) \rightarrow (\sigma|j,\beta,R \cup \{j,r,i\}) \\
&\textsc{RightArc} && (\sigma|i|j,\beta,R) \rightarrow (\sigma|i,\beta,R \cup \{i,r,j\}) \\
&\textsc{Shift} && (\sigma,i|\beta,R) \rightarrow (\sigma|i,\beta,R)
\end{aligned}$$

Association scores can inform a parser about whether an attachment with a particular dependency relation should be made between two attachment sites. For each parser state, two attachments are possible with any of the dependency relations that are available in that system.[1] Association scores for all possible attachments provide disambiguation cues at each state. They are added to the feature vector that is used as input to the transition classifier. Association score vectors enhance existing vector representations of words, part-of-speech tags, characters, dependency relations and morphological features.

2.2 Parser Integration

For each parser state, association scores are retrieved for LEFTARC and RIGHTARC transitions, and for all possible dependency relations. Equation 1 defines the association score vector for a stack-projective

[1] A third option is to apply a SHIFT transition which does not introduce an attachment.

transition system with transitions between the token on top of the stack s_0 and the second token on the stack s_1:

$$v_{assoc} = [assoc(s_0, s_1, r),\ assoc(s_1, s_0, r) \mid \forall r \in R] \tag{1}$$

Example 2 provides the resulting association score vector in a stack-projective system with a dependency relation set that contains the *subject*, *object* and *preposition* relation.

$$\begin{aligned} v_{assoc} = [&assoc(s_0, s_1, subject),\ assoc(s_0, s_1, object),\ assoc(s_0, s_1, preposition),\\ &assoc(s_1, s_0, subject),\ assoc(s_1, s_0, object),\ assoc(s_1, s_0, preposition)] \end{aligned} \tag{2}$$

$$\text{with } R = \{subject, object, preposition\}$$

If no association score is available for a dependency triple, a default is assigned. An optional binary indicator $b \in \{0, 1\}$ specifies whether the dependency triple was known. This makes it possible for the model to distinguish between the default value and association strengths that overlap with the default value. The binary indicators are added to the association score vector. The association score vectors are concatenated with the remaining input feature vectors to represent a parser configuration.[2]

2.3 Association Metric Variants

Pointwise mutual information. Traditionally, PMI has been a means to capture bilexical preferences. Normalized (NPMI, Bouma (2009)) and positive normalized PMI (PNPMI, Van de Cruys (2011)) with add-1 Laplace smoothing have been tested in the parsing model. Given the dependency triple $h \xrightarrow{r} d$, consisting of the head h, dependent d and dependency relation r, PMI is defined as:

$$PMI(h \xrightarrow{r} d) = log \frac{p(h \xrightarrow{r} d)}{p(h \xrightarrow{r})\ p(\xrightarrow{r} d)} \tag{2}$$

The probability of h and d as heads and dependents with relation r is represented as $p(h \xrightarrow{r})$ and $p(\xrightarrow{r} d)$, the dependency triple probability as $p(h \xrightarrow{r} d)$. Normalized PMI

$$PMI_{norm}(h \xrightarrow{r} d) = \frac{PMI(h \xrightarrow{r} d)}{-log\ p(h \xrightarrow{r} d)} \tag{3}$$

is a more easily interpretable variant of PMI, limiting the range of PMIs to lie between -1 and 1. Positive PMI

$$PMI_{pos}(h \xrightarrow{r} d) = max(PMI(h \xrightarrow{r} d), 0) \tag{4}$$

rounds negative PMIs to 0.

Dependency embedding scores. PMI is likely to suffer from sparseness of dependency triples in the training data. Previous attempts have used back-off models (Collins and Brooks, 1995) to counteract this problem. The dependency embedding model by Levy and Goldberg (2014a) estimates probabilities for unseen triples $h \xrightarrow{r} d$ from word embeddings. The model predicts the probability $p(1|h \xrightarrow{r} d)$ of a dependency triple. Words are represented as embeddings that are trained jointly with the classifier $p(1|h \xrightarrow{r} d)$.

An embedding-based association score for the head word embedding W_h and the context embedding $C_{d,r}$ of the dependent d that is related to a head h via the dependency relation r can be formulated as:

$$assoc_{dep}(h \xrightarrow{r} d) = p(1|h \xrightarrow{r} d) = \sigma(W_h \cdot C_{d,r}) \tag{5}$$

where $C \in \mathbb{R}^{|V| \times r \times d}$ and $W \in \mathbb{R}^{|V| \times d}$. In the current model, the maximum entropy probability of 0.5 is assigned as a default when no embedding for h, d or both is available and no score can be calculated. Further model variations also include a binary indicator to distinguish the default score from a calculated embedding-based score. In a more finegrained binary indicator model, the indicator informs the parser

[2] A complete list of parser input features can be found in Appendix A.

for which of the two tokens no embedding was available.

Levy and Goldberg (2014b) have shown that the skip-gram model is an implicit factorization of the shifted PMI matrix of word co-occurrences. Dependency embeddings (Levy and Goldberg, 2014a) therefore implicitly factorize the shifted PMI matrix of head-dependent co-occurrences. Hence, association scores based on dependency embeddings (Kiperwasser and Goldberg, 2015) can be seen as correlated with PMIs.

3 Experiments

3.1 Experimental Setup

The neural transition-based dependency parser of De Kok and Hinrichs (2016) serves as the baseline for the experiments. Words, part-of-speech tags and characters are represented as vectors that were trained with structured skip-gram (Ling et al., 2015). Topological fields are used as additional input features. The parser does pseudo-projective parsing (Nivre and Nilsson, 2005) and was trained on the shuffled TüBa-D/Z (Telljohann et al., 2017) that contains 105K sentences and 1.9M tokens of manually labeled data from the Berliner Tageszeitung (taz). Non-gold part-of-speech tags were trained via 10-fold jackknifing on the TüBa-D/Z.[3] The data was split in a 7:1:2 ratio for respectively training, development and testing. Association scores are retrieved for lowercased word forms to increase lexical coverage. Common and proper nouns are typically capitalized in German and were therefore not lowercased.

Results are presented as labeled (LAS) and unlabeled attachment scores (UAS) including punctuation. Accuracies for inversion and prepositions indicate performance on resolving ambiguities. Inversion accuracy reports correct labeling of subjects and objects in clauses with fronted object. Preposition accuracy comprises all correct heads and labels of prepositional phrases and objects. The test set contains 1,887 cases of inversion (5.82 percent of all clauses) and 31,687 prepositional phrases and objects.

3.2 PMIs in Neural Dependency Parsing

A table of PMIs was generated for dependency triples $h \xrightarrow{r} d$ from the German newspaper taz (393.7M tokens, 22.8M sentences) and a dump of the German Wikipedia from January 2018 (803.5M tokens, 39.9M sentences), two subcorpora of the TüBa-D/DP treebank (De Kok and Pütz, 2019) parsed by the De Kok and Hinrichs (2016) parser without association scores. All dependency triples not contained in the table are mapped to the most neutral value of 0. The PMI table is generated once in linear time. The same holds for the dependency embeddings described in Section 3.3. Each association score retrieval is then done in constant time so that the linear time property of parsing remains unchanged.

Model	**LAS**	**UAS**	**Inversion accuracy**	**Preposition accuracy**
De Kok and Hinrichs (2016)	92.01	93.88	81.03	77.80
+ NPMI, minfreq 5	**92.27**	**94.01**	81.93	78.60
+ NPMI, minfreq 50	92.14	93.92	82.25	78.29
+ NPMI, minfreq 100	92.16	93.92	80.72	78.56
+ NPMI, minfreq 5, binary	92.18	93.94	**82.57**	**78.78**
+ NPMI, minfreq 50, binary	92.16	93.93	81.93	78.35
+ NPMI, minfreq 100, binary	92.18	93.96	81.67	78.29
+ PNPMI, minfreq 5	92.21	93.99	82.09	78.44
+ PNPMI, minfreq 50	92.19	93.95	81.46	78.66
+ PNPMI, minfreq 100	92.17	93.94	82.25	78.57

Table 2: Parser accuracy (overall, inversion, preposition attachment) for neural dependency parsing with PMI-based association scores. The NPMI model with minimum frequency 5 achieves the best overall performance.

[3]Using the *sticker* software package: https://github.com/danieldk/sticker.

PMI models with minimum dependency triple frequencies of 5, 50 and 100 have been trained with both NPMI and PNPMI scores. NPMI models have been tested with and without binary indicator. Results for the PMI models are given in Table 2.

The best PMI model uses normalized PMI with a minimum frequency of 5. The model outperforms the baseline by 0.26 LAS points which is significant in the Wilcoxon test (Dror et al., 2018) with $p < 5.24 \times 10^{-10}$. It also improves the LAS by 0.03 points over the best embedding-based model but the improvement is not statistically significant. Larger improvements can be seen for both sorts of ambiguity. The best model increases inversion LAS by 1.54 points and preposition LAS by 0.98 points over the baseline.

3.3 Dependency Embedding Scores in Neural Dependency Parsing

For the embedding-based model, dependency embeddings with 300 dimensions were trained with the algorithm from Levy and Goldberg (2014a).[4] Different embeddings have been trained on pseudo-projectivized and non-projective versions of taz, Wikipedia, and the German europarl (1.25B tokens and 42.1M sentences in total). The number of dependency relations varies from 38 non-projective to 212 pseudo-projective relations.

All embedding variants have been trained on regular head-dependent and inverse dependent-head relations. A fully typed model was trained on context that includes the token typed per dependency relation. A second semi-typed model includes the token without dependency relations as context. For both models, variants with and without binary indicator have been evaluated. The binary model uses a simple binary indicator which labels association scores as default or as being calculated from dependency embeddings. A more finegrained triple-binary model for fully typed embeddings evaluates the following three conditions to true or false: 1) the head word embedding could be retrieved from the focus matrix, 2) the dependent word embedding, i.e. the combination of the context token and the dependency relation, could be retrieved from the context matrix, 3) an embedding for the context token could be retrieved from the focus word matrix, indicating whether there exists a word embedding for the token at all. The double-binary model for semi-typed embeddings indicates whether an embedding has been found for the focus and the context token. As the context token is not typed for dependencies in the semi-typed model, the context matrix contains entries for tokens without the different dependency relations they occur with.

Model	**LAS**	**UAS**	**Inversion accuracy**	**Preposition accuracy**
De Kok and Hinrichs (2016)	92.01	93.88	81.03	77.80
+ projective, fully typed	92.23	**93.97**	82.57	78.55
+ projective, fully typed, binary	**92.24**	**93.97**	**83.36**	78.47
+ projective, fully typed, triple-binary	92.16	93.88	**83.36**	**78.62**
+ projective, semi-typed	92.11	93.94	80.66	77.99
+ projective, semi-typed, binary	91.98	93.89	80.61	77.71
+ projective, semi-typed, double-binary	92.07	93.93	81.93	77.98
+ non-projective, fully typed	92.17	93.93	81.46	78.17
+ non-projective, fully typed, binary	92.22	**93.97**	82.20	78.45
+ non-projective, fully typed, triple-binary	92.08	93.86	82.99	78.26

Table 3: Parser accuracy (overall, inversion, preposition attachment) for neural dependency parsing with embedding-based association scores. The overall best model uses projectivized, fully typed dependency embeddings with a binary indicator.

Results for parsing with association scores based on dependency embeddings are shown in Table 3. The overall best embedding-based model uses projectivized, fully typed embeddings with a binary indicator. The model outperforms the baseline parser by 0.23 LAS points, significant in the Wilcoxon

[4]Using the *finalfrontier* software package: https://finalfusion.github.io/finalfrontier.

test ($p < 1.94 \times 10^{-7}$), and remains 0.03 points below the best PMI model. Embedding-based models are only superior to PMI models when it comes to inversion LAS. There, the best embedding-based model improves by 2.33 points over the baseline, compared to 1.54 points improvement of the best PMI model.

4 Evaluation

Both the PMI-based and embedding-based models perform better than the baseline. Overall performance will improve by more correctly solved ambiguous attachments. Lexical associations between more than two tokens may be necessary to further improve ambiguity resolution. For PP attachment, the compatibility between the preposition, its modifier noun and the verbal or nominal head candidate of the PP have to be modeled. De Kok et al. (2017) have shown that trilexical preferences help to better capture attachment preferences of the preposition.

It can also be beneficial to make competing attachment sites available to the parser. Currently, association scores are only computed for the two attachment candidates for any given parser state. With beam search, several attachment candidates can compete in different analyses. The best candidate can then be chosen from all or the *n* best candidates (Zhang and Clark, 2008; Andor, 2016).

5 Ambiguity Resolution with Association Metrics

Most parser errors still involve a limited number of dependency relations, as shown in Table 1. Errors in PP attachment, subjects and objects often can be traced back to problems with resolving ambiguities. An evaluation of association scores for particular word pairs can show if such scores can be useful in parsing ambiguous sentences. Table 4 lists PMI- and embedding-based scores for selected word pairs and dependency relations. Random pairs that are common in everyday language are distinguished from pairs that occur in subject-object inversion and have been incorrectly attached by the (best-performing embedding-based) parser. PMIs have been retrieved from the positive normalized PMI table with minimum frequency 5. Embedding-based scores were calculated from projectivized, fully typed dependency embeddings.

	PNPMI		**Embedding-based**		**Example**
Relation	*Subject*	*Object*$_{acc}$	*Subject*	*Object*$_{acc}$	
Random pairs					
isst, sie	–	0.0617	0.9778	0.9863	Sie isst Spaghetti.
isst, Spaghetti	–	–	0.1375	0.9996	*'She eats Spaghetti.'*
trinkt, Mann	0.1341	–	0.9883	0.8776	Der Mann trinkt Milch.
trinkt , Milch	–	0.3627	0.9509	0.9997	*'The man drinks milk.'*
weiß, Computer	–	–	0.9280	0.1397	Ein Computer weiß alles.
weiß, alles	–	0.1995	0.9847	0.9948	*'A computer knows everything.'*
Incorrectly attached inversion pairs					
erstatteten, Angeklagten	–	–	0.9917	0.9545	Strafanzeigen erstatteten die Angeklagten
erstatteten, Strafanzeige	0.4645	0.5604	0.9566	0.9996	*'The defendants pressed criminal charges'*
wollte, niemand	0.1906	0.1750	0.9940	0.9366	Nur wollte den Krempel niemand.
wollte, Krempel	–	–	0.5458	0.0008	*'But nobody wanted that junk.'*
tragen, Studierenden	0.0794	–	0.9645	0.7761	Das Risiko tragen die Studierenden.
tragen, Risiko	0.0825	0.2540	0.9269	0.9972	*'The students take the risk.'*

Table 4: PMI and embedding-based scores for random and incorrectly attached dependency triples.

Problems of data sparsity can indeed be solved by using embedding-based rather than PMI-based scores, as Table 4 shows. In spite of a low frequency threshold of 5, the PMI table is very sparse compared to the embedding-based scores. However, when a PMI is available scores indicate the correct tendency in the majority of the cases. Considering that all unknown values are equal to the default PMI of 0.0, the tendencies are correct for e.g. *trinkt* 'drinks' which prefers to attach *Mann* 'man' as the subject and *Milch* 'milk' as the direct object. The tendencies of embedding-based scores are mostly correct, such as the preference of *Spaghetti* 'spaghetti' to attach to *isst* 'eats' as a direct object. Wider

lexical coverage of embedding-based models may not lead to any gains over PMI-based models partially due to the architecture of the neural dependency parser which already encodes information about co-occurrences in the distributional representations of the input tokens.

6 Conclusion

This paper presented a technique to include association metrics into a neural transition-based dependency parser for German. PMI and embedding-based association scores have been tested. Both PMI-based and embedding-based models significantly outperform the baseline. In spite of the wider lexical coverage of embedding-based models, PMI models achieve accuracies on a par with embedding-based models.

A qualitative analysis revealed that association scores in parts provide useful disambiguation cues to the parser. Follow-up experiments in other languages with relatively free word order and moderately complex morphology will further investigate the effect of association metrics on neural transition-based dependency parsing. Due to its similarity to German, Dutch will be the first language to be examined. Trilexical rather than bilexical preferences could further improve results. Keeping more competing attachment candidates through beam search is another promising direction for future work. As an alternative to association scores, a compatibility model that is directly integrated into the parser could be considered.

Acknowledgments

Financial support for the research reported in this paper was provided by the German Research Foundation (DFG) as part of the Collaborative Research Center "The Construction of Meaning" (SFB 833), project A3.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally Normalized Transition-Based Neural Networks. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics: Volume 1, Long Papers*:2442–2452. Berlin, Germany.

Gerlof Bouma. 2009. Normalized (Pointwise) Mutual Information in Collocation Extraction. *From Form to Meaning: Processing Texts Automatically, Proceedings of the Biennial GSCL Conference 2009*:31–40. Tübingen, Germany.

Danqi Chen and Christopher D. Manning. 2014. A Fast and Accurate Dependency Parser Using Neural Networks. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*:740–750. Doha, Qatar.

Kenneth Ward Church and Patrick Hanks. 1990. Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics*, 16(1):22–29.

Michael Collins and James Brooks. 1995. Prepositional Phrase Attachment through a Backed-Off Model. *Proceedings of the Third Workshop on Very Large Corpora*:27–38. Cambridge, MA.

Daniël de Kok and Erhard Hinrichs. 2016. Transition-Based Dependency Parsing with Topological Fields. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics: Volume 2, Short Papers*:1–7. Berlin, Germany.

Daniël de Kok, Jianqiang Ma, Corina Dima, and Erhard Hinrichs. 2017. PP Attachment: Where do We Stand? *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*:311–317. Valencia, Spain.

Daniël de Kok and Sebastian Pütz. 2019. Tüba-D/DP Stylebook. *https://github.com/sfb833-a3/tueba-ddp/blob/master/stylebook/stylebook-r4.pdf* [last visited on 05/23/2019].

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. *International Conference on Learning Representations*. Toulon, France.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The Hitchhiker's Guide to Testing Statistical Significance in Natural Language Processing. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics: Volume 1, Long Papers*:1383–1392. Melbourne, Australia.

Robert Mario Fano. 1961. *Transmission of Information: A Statistical Theory of Communications.* MIT Press, Cambridge, MA.

Donald Hindle and Mats Rooth. 1993. Structural Ambiguity and Lexical Relations. *Computational Linguistics*, 19(1):103–120.

Eliyahu Kiperwasser and Yoav Goldberg. 2015. Semi-Supervised Dependency Parsing Using Bilexical Contextual Features from Auto-Parsed Data. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*:1348–1353. Lisbon, Portugal.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing.* Morgan and Claypool, San Rafael, CA.

Omer Levy and Yoav Goldberg. 2014a. Dependency-Based Word Embeddings. *Proceedings of the 52nd Annual Meeting for the Association for Computational Linguistics, Volume 2: Short Papers*:302–308. Baltimore, MD.

Omer Levy and Yoav Goldberg. 2014b. Neural Word Embeddings Implicit Matrix Factorization. *Advances in Neural Information Processing Systems: Annual Conference on Neural Information Processing Systems 2014*:2177–2185.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/Too Simple Adaptations of Word2Vec for Syntax Problems. *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*:1299–1304. Denver, CO.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of Workshop at the International Conference on Learning Representations 2013*:1–12. Scottsdale, AZ.

Seyed Abolghasem Mirroshandel and Alexis Nasr. 2016. Integrating Selectional Constraints and Subcategorization Frames in a Dependency Parser. *Computational Linguistics*, 42(1):55–90.

Joakim Nivre. 2004. Incrementality in Deterministic Dependency Parsing. *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*:50–57. Barcelona, Spain.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-Projective Dependency Parsing. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*:99–106. Ann Arbor, MI.

Adwait Ratnaparkhi. 1998. Statistical Models for Unsupervised Prepositional Phrase Attachment. *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*:1079–1085. Montréal, Canada.

Heike Telljohann, Erhard Hinrichs, Sandra Kübler, Heike Zinsmeister, and Kathrin Beck. 2017. Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z). *http://www.sfs.uni-tuebingen.de/fileadmin/static/ascl/resources/tuebadz-stylebook-1707.pdf* [last visited on 05/08/2019].

Tim van de Cruys. 2011. Two Multivariate Generalizations of Pointwise Mutual Information. *Proceedings of the Workshop on Distributional Semantics and Compositionality*:16–20. Portland, OR.

Gertjan van Noord. 2007. Using Self-Trained Bilexical Preferences to Improve Disambiguation Accuracy. *Proceedings of the 10th Conference on Parsing Technologies*:1–10. Prague, Czech Republic.

Martin Volk. 2002. Combining Unsupervised and Supervised Methods for PP Attachment Disambiguation. *Proceedings of the 19th International Conference on Computational Linguistics, Volume 1*:1–7. Taipei, Taiwan.

Yue Zhang and Stephen Clark. 2008. A Tale of Two Parsers: Investigating and Combining Graph-Based and Transition-Based Dependency Parsing Using Beam-Search. *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing: Volume 2, Short Papers*:562–571. Honolulu, HI.

Appendix A. Parser Inputs

The parser uses vector representations of the word (TOKEN), its part-of-speech tag (TAG), dependency relation (DEPREL), characters (CHAR) and topological field (TF) as inputs. Different positions on the stack and buffer are addressed for each feature. The full list of features is provided below. [BUFFER 0], for example, refers to the first token on the buffer, [STACK 0, LDEP 0] addresses the leftmost dependent of the token on top of the stack. Character representations are included for the word prefix and suffix each of length 4.

[STACK 0] TOKEN
[STACK 1] TOKEN
[STACK 2] TOKEN
[STACK 3] TOKEN
[BUFFER 0] TOKEN
[BUFFER 1] TOKEN
[BUFFER 2] TOKEN
[STACK 0, LDEP 0] TOKEN
[STACK 1, LDEP 0] TOKEN
[STACK 0, RDEP 0] TOKEN
[STACK 1, RDEP 0] TOKEN

[STACK 0] TAG
[STACK 1] TAG
[STACK 2] TAG
[STACK 3] TAG
[BUFFER 0] TAG
[BUFFER 1] TAG
[BUFFER 2] TAG
[STACK 0, LDEP 0] TAG
[STACK 1, LDEP 0] TAG
[STACK 0, RDEP 0] TAG
[STACK 1, RDEP 0] TAG

[STACK 0] DEPREL
[STACK 0, LDEP 0] DEPREL
[STACK 1, LDEP 0] DEPREL
[STACK 0, RDEP 0] DEPREL
[STACK 1, RDEP 0] DEPREL

[STACK 0] CHAR 4 4
[STACK 1] CHAR 4 4
[BUFFER 0] CHAR 4 4

[STACK 0] TF
[STACK 1] TF
[STACK 2] TF
[STACK 3] TF
[BUFFER 0] TF
[BUFFER 1] TF
[BUFFER 2] TF