Parsing Morphologically Rich Languages with (Mostly) Off-The-Shelf Software and Word Vectors

Arne Köhn
Department of Informatics
Universität Hamburg
koehn@informatik.uni-hamburg.de

U Chun Lao and **AmirAli B. Zadeh** and **Kenji Sagae**
Institute for Creative Technologies
University of Southern California
ulao@usc.edu, {zadeh,sagae}@ict.usc.edu

Abstract

As a contribution to the 2014 SPMRL shared task on parsing morphologically rich languages, we show that it is now possible to achieve high dependency accuracy using existing parsers without the need for intricate multi-parser schemes even if only small amounts of training data are available. We further show that the impact of using word vectors on parsing quality heavily depends on the amount of morphological information that is available. In addition, we discuss the use of parser scores for selection of morphological lattice paths, showing that there is much discriminative power in syntactic parsers for morphological disambiguation.

1 Introduction

Results from the 2013 shared task on Statistical Parsing of Morphologically Rich Languages (SPMRL) have shown that it is now possible to obtain high dependency accuracy on a variety of morphologically rich languages. The best results for every one of the nine languages included in the 2013 SPMRL shared task were obtained by Björkelund et al. (2013), who designed a very effective but intricate ensemble approach involving reranking of merged n-best lists from three parsers. We describe our contribution to the 2014 SPMRL shared task, focusing on the use of a single existing parser without extensive customization, addressing the question of what level of accuracy can be expected from off-the-shelf software currently available.

We also investigate the use of vector representations of words in statistical parsing of morphologically rich languages, which constitutes a form of semi-supervised learning, since these vector representations are derived from large unlabeled corpora. Using word vectors as features for syntactic parsing has been challenging because most parsers only use discrete features. Vectors are often used, instead, to create word clusters, from which features can be derived (a notable exception is Collobert and Weston (2008)). In recent work, Lei et al. (2014) proposed the use of low-rank tensors for scoring in addition to the scoring elements used by Martins et al. (2013). This approach allows vectors to be incorporated directly as continuous features, and is the basis of our experiments with word vectors.

In the first half of this paper, we explore the effects of using different kinds of word vectors with a higher-order model in the state-of-the-art dependency parser of Lei et al. (2014), comparing the results to those obtained by the same parser without word vectors. We also investigate the possible overlap in the information contained in morphological annotation and in word vectors by considering settings where gold standard morphological information is available, and where no morphological information is available. In the second half of the paper, we present a separate set of experiments using the Hebrew dataset, where we consider the more realistic setting of automatic prediction of part-of-speech and morphological features, and attempt to determine whether the use of scores assigned by a discriminative parser are effective for

	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish	Average
Turbo	71.16	81.34	75.37	75.90	81.38	64.58	64.91	69.81	73.06
Mate	74.75	81.22	88.23	87.56	68.29	79.61	88.14	80.73	81.07
RBG	88.74	92.32	92.22	88.94	88.05	87.52	92.52	85.06	89.42

Table 1: Comparison of unlabeled accuracy results obtained with TurboParser, Mate parser and RBGParser.

scoring of paths in lattices encoding morphological information. All of our experiments were conducted on the limited labeled data track (5k), where training sets for each language consisted of 5,000 labeled sentences.[1]

2 Parsing with RBGParser and the 5k datasets

In our experiments we used three off-the-shelf parsers, which we trained using the 5k data sets for each language in the SPMRL 2014 shared task. The primary parser we used to generate our results is RBGParser (Lei et al., 2014), which has been released recently and has been shown to have excellent performance on the CoNLL 2006 multilingual dependency parsing task (Buchholz and Marsi, 2006). To gauge the parsing accuracy of RBGParser when only given small amounts of training data, we compare its results against TurboParser (Martins et al., 2013) and the Mate parser (Bohnet, 2010)[2]. Lei et al. (2014) reported an increase of 0.3 percentage points against TurboParser on average on the CoNLL 2006 dataset (Buchholz and Marsi, 2006). The results for the 5k training sets are much more pronounced, as can be seen in Table 1. The large gap in accuracy between RBGParser and both Mate parser and TurboParser and the differences between these two parsers are especially interesting since Foth et al. (2014) evaluated parsing accuracy as a function over the training size on the Hamburg Dependency Treebank and both parsers performed well with a training size of only 1,000 sentences.

2.1 Word Vectors

There are several ways to create word embeddings in a high dimensional space. We consider skipgrams and continuous bag-of-words (Mikolov et al., 2013), the two variants implemented in word2vec[3], which we used to carry out our experiments. The continuous bag-of-words model is designed to predict the current word given its context, and the skipgram model is designed to predict each of the words in the context given the current word.

In our experiments, we examined the performance of three different types of vectors for each language (each with a window size of five words):

- Skipgram with five negative examples (200 dimensions, negative sampling);
- Continuous bag-of-words (200 dimensions, hierarchical softmax);
- A concatenation of the two above (400 dimensions).

All word vectors were trained on the unlabeled data for each language provided for the shared task. RBGParser was trained on the 5k gold training set for each language[4] with each set of word vectors, as well as without word vectors.

The improvements observed due to word vectors (Table 2) are generally similar to those reported by Lei et al. (2014). However, for some languages, the types of word vectors we used did not impact the accuracy achieved, and results were no better than those obtained without word vectors.

[1]The software and settings for our experiments are available on `http://nats-www.informatik.uni-hamburg.de/User/ArneKoehn/spmrl14`

[2]The results reported here are based on the development sets of the respective languages, unless otherwise noted.

[3]`https://code.google.com/p/word2vec/`

[4]We excluded Arabic because the memory requirements for training the RBGParser with the 5k training set, over 40Gb, exceeded the resources available to us at the time of the experiments.

Word Vector	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
None	88.74	92.32	92.22	88.94	88.05	87.52	92.52	85.06
CBOW	89.05	92.33	**92.30**	89.40	**88.43**	87.51	**92.60**	**85.29**
Skipgram	**89.15**	**92.43**	92.22	**90.57**	88.38	**87.73**	92.26	85.11
Combined	89.00	92.23	92.29	89.13	88.39	86.87	92.48	85.08

Table 2: Unlabeled parsing accuracy obtained with different word vector representations.

Word Vector	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
IMS-SZEGED	90.18	90.63	89.70	88.95	90.68	84.78	93.11	89.11
RBGParser	89.48	90.45	89.34	88.09	89.33	85.99	89.83	88.90
difference	-0.70	-0.18	-0.36	-0.86	-1.35	+1.21	-3.28	-0.21

Table 3: Unlabeled parsing accuracy comparison of RBGParser with skipgram word vectors versus the best performing system of SPMRL 2013 (IMS-SZEGED), evaluated on the test sets.

Table 3 shows the official results on the test sets compared to the best performing parser of last year, Björkelund et al. (2013). As can be seen, using a single parser instead of a parser ensemble only incurs slight degradation in parsing performance.

2.2 Parsing without Morphological Annotation

A possible explanation for the lack of larger accuracy improvements when using word vectors is that the information they encode is also present in the morphological information provided while training and parsing. To test this hypothesis, we stripped all morphologic information (i.e. the morphological features and the part-of-speech column) from both the training and test sets. The results obtained when parsing without morphological information show that indeed the word vectors make a difference of five to seven percentage points in this scenario, showing that word vectors do contribute substantially in the absence of part-of-speech and morphological information (see Table 4). They cannot, however, offset the loss incurred by not having gold standard morphological annotation.

2.3 Relabeling

RBGParser, as well as TurboParser, decides on the label of each possible edge before actually parsing. This way the parsing problem is less complex because considering all possible labels for each edge while parsing would multiply the number of possible parses for a sentence by l^{n-1}, where l is the number of labels in the dependency scheme and n is the number of words in the input. However, the labeling may benefit from structural features beyond the identity of the head and dependent of an edge. Therefore, we implemented a simple labeler which relabels a dependency tree using the features listed in Table 5. The labeler employs a multiclass averaged perceptron and consistently outperforms the labeling accuracy of RBGParser (Table 6).

	Basque	French	German	Hebrew	Hungarian	Swedish
With vectors	83.68	82.75	84.61	82.37	80.73	76.81
Without vectors	78.88	79.60	79.53	77.61	73.62	69.95

Table 4: Unlabeled accuracy results obtained when parsing without any morphological annotation, with and without word vectors.

Words used	Features
(dependent, head)	direction, morph
dependent	word, pos, cpos, morph, morph splitted, morph×(pos cpos)
head	word, pos, cpos, morph, morph×(pos cpos)
dep[-1]	word, pos
dep[-2]	word, pos
dep[1]	word, pos
dep[2]	word, pos

Table 5: Features used by the labeler, words and features in brackets are combined, dep[x] is the x^{th} word right of the dependent.

	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
original	80.98	90.93	93.11	86.60	83.71	94.61	89.44	82.56
relabel	88.27	94.74	94.53	87.43	90.62	96.95	92.04	83.10

Table 6: Labeling performance of our simple labeler versus the original RBGParser labeling.

3 Parsing with Lattices

The experiments and results presented so far, including those where part-of-speech and morphological features were explicitly removed from the input, assumed fixed tokenization (gold standard word segmentation, specifically) and no ambiguity in the part-of-speech and morphology analyses for the input sentences. In more realistic scenarios, however, better results from a full parsing pipeline may be obtained when ambiguity is considered, and different lexical analyses may result in different word segmentation options of the input. Ambiguities in word segmentation, morphology and part-of-speech can be encoded efficiently in a lattice, which is a directed acyclic graph with a source node that serves as a starting point, from which paths through the lattice correspond to individual readings or analyses for the entire input sentence. Given the potentially large ambiguity represented in the lattice, selecting a correct path (analysis) through lattices is often essential for an accurate parsing result.

Of the nine datasets provided, only the Hebrew dataset includes lattices encoding ambiguities in tokenization, part-of-speech and morphology, so our experiments are focused on this one dataset. An automatically disambiguated path is provided, in addition to a gold standard analysis.

3.1 Preliminary Evaluation of Lattice Quality

Because, as explained in Seddah et al. (2013), the Hebrew lattices may not contain the gold standard paths, we first examined the upper-bound for our experiments based on these lattices. To measure this upper-bound, we use the gold standard analyses to create an oracle to select the best possible path for each sentence (the “oracle best” path).

The metric used in the evaluation on all lattice path sets is the F-score of tagged tokens, which requires segmentation, the part-of-speech tag, and all morphological features to match a gold standard token for the predicted token to be considered correct. The F-score for the oracle best path is 0.60, which is below the F-score of the disambiguated path provided with the dataset, 0.62. This means that even with a perfect disambiguation model, paths extracted from the lattices provided would be of lower quality than the disambiguated paths provided, which were extracted from a different, larger lattice. As a result, there is little hope that parsing results obtained with the provided lattices would be better than results obtained with the provided disambiguated paths. However, due to time constraints, we did not examine parsing accuracy resulting from the use of lattices, and instead investigated only the token F-scores of extracted paths, with the reasonable assumption that higher token F-scores correspond to higher parsing accuracy. Validation of this assumption is left as future work.

3.2 A Simple Path Selection Approach

As a simple way of selecting a path through the lattice, we trained a logistic regression model on the set of edges extracted from both the lattices and the gold standard analyses using simple features that include word forms, part-of-speech tags, and morphological features for two adjacent tokens. The model was trained locally by simply taking gold standard edges as positive examples and other edges as negative examples, and only edges that have at least one of its ends as part of the gold standard path are included in training. We used the resulting edge scoring model to select a path automatically, and to select k unique paths for each sentence using beam search. In our experiments we extracted 100 paths for each sentence from the dev set. The F-score of the highest scoring path is 0.46, well below the oracle best path's F-score of 0.60. While it is possible that a better model would achieve a higher F-score, simply maximizing F-score was not the main goal of our experiments, since the upper-bound is still lower than the disambiguation provided in the dataset. Instead, we use this model to generate multiple candidate paths, and attempt to determine whether a parser model can be used to determine the quality of the candidate paths.

3.3 Path Selection with TurboParser

Syntactic parsing can be (and often is) formulated as finding the best scoring parse p for a given sentence s and a scoring function f:

$$p = \arg\max_{p' \in P} f(p', s)$$

However, sometimes the segmentation of a sentence into words is not unique and the correct segmentation and morphologic annotation needs to be found. Given the fact that some segmentations and annotations will violate syntactic constraints, we can try to find the best segmentation by choosing the one that has the highest scoring parse:

$$s = \arg\max_{s' \in S}(\max_{p \in P} f(p, s'))$$

TurboParser (Martins et al., 2013) translates the parsing problem into an integer linear program and solves it using a LP relaxation. The optimal parse then ideally is the highest scoring admissible solution for the ILP. A parse will have a bad score (i.e. a high log potential) if it is not a good fit for the sentences. We hypothesize that this also works the other way round: if the sentence is ungrammatical, we can hope that there is no fitting parse with low log-potential. Therefore, the best parse of an ungrammatical sentence should have a worse score than the best parse of a grammatical one.

To verify this hypothesis, we performed several experiments on the beam search selected 100 best paths from the Hebrew lattice. Using the score generated by TurboParser on each path, we can rerank these paths and choose one best path for each sentence. These best paths acceding to TurboParser scores have F-score of 0.48, which is slightly higher than the F-score obtained with our simple path selection model.

Since the F-score obtained from paths selected by TurboParser is limited by the logistic regression model that generates the 100-best list from which TurboParser selects its highest scoring path, we ran an additional experiment where the oracle best path is added to the list of 100-best paths, if it is not already included. This experiment is intended to provide some indication of whether TurboParser might perform better in path selection given longer n-best lists. Of course, the use of arbitrarily long n-best lists is impractical, since each path in the n-best list is parsed individually, meaning that a 1000-best list would make parsing 1000 times slower. With the 100-best paths selected by our simple selection model plus the oracle best path, the paths selected by TurboParser have an F-score of 0.57, which is close to the upper-bound established by the oracle best path, 0.60. This serves as indication that TurboParser may in fact be effective in selecting lattice paths.

Finally, we ran another experiment where instead of adding the oracle best path to the list of 100-best paths, we add the gold standard analysis to the list of 100-best paths. This no longer approximates any realistic path selection problem, since the gold standard path is often not in the lattices used in the experiment. However, with this experiment we explore whether TurboParser can distinguish an analysis of substantially higher quality from those contained in the lattice. When given lists containing 100-best

Path selection scheme	F-score
disambiguated path provided with dataset	0.6204
oracle best path	0.6045
predicted path	0.4645
TurboParser selection from top 100 scoring paths	0.4769
TurboParser selection from top 100 scoring paths + oracle path	0.5679
TurboParser selection from top 100 scoring paths + gold standard path	0.9070

Table 7: Comparison of various lattice path selection schemes.

paths extracted from the lattice plus the gold standard analysis, TurboParser selections result in an F-score of 0.91, confirming that TurboParser can effectively select the higher quality analysis. A summary of our results on lattice path selection is shown in Table 7.

This preliminary investigation of whether a parsing model is effective in selecting paths through lattices that encode ambiguities in word segmentation, part-of-speech and morphology leaves out two important questions: how well this type of selection would perform with higher quality lattices (and higher quality n-best lists), and how well the token F-scores used in our experiments correlate with parsing accuracy. Addressing both of these questions is left as future work.

4 Conclusions

We demonstrated that a single off-the-shelf parser can be trained with no other customization to produce high accuracy dependency parsing results. RBGParser, which has been released recently, performed very well across eight of the nine languages in the SPMRL 2014 shared task (we were unable to train a model for Arabic), with high unlabeled accuracy. The addition of a simple labeler produces high labeled dependency scores. RBGParser outperforms both TurboParser and the Mate parser when trained on the 5k training sets, and can produce good results even when no morphological analysis is available. In that case the parser greatly benefits from the use of distributed vector representations of words derived from unlabeled data.

In our experiments, we compared only the two extreme cases where no morphological information is available and where gold standard morphological information is available, leaving out the more practical scenario where automatic morphological analysis produces reasonable results. Investigation of the use of word vectors under that condition is left as future work.

Finally, we presented preliminary results that suggest that TurboParser is capable of selecting high quality morphological analyses, but we have not quantified the effect of this type of selection on parse accuracy. In future work, it would be interesting to examine whether RBGParser's superior accuracy can be combined with a parser-based path selection scheme without the need for parsing sentences multiple times.

Acknowledgments

We express our gratitude to the treebank providers for each language: Arabic (Maamouri et al., 2004; Habash and Roth, 2009; Habash et al., 2009), Basque (Aduriz et al., 2003; Aldezabal et al., 2008), French (Abeillé et al., 2003; Candito et al., 2010) , Hebrew (Sima'an et al., 2001; Tsarfaty, 2010; Goldberg, 2011; Tsarfaty, 2013), German (Brants et al., 2002; Seeker and Kuhn, 2012), Hungarian (Csendes et al., 2005; Vincze et al., 2010), Korean (Choi et al., 1994; Choi, 2013), Polish (Woliński et al., 2011; Świdziński and Woliński, 2010; Wróblewska, 2012), and Swedish (Nivre et al., 2006).

We also would like to thank the following institution for providing the unlabeled data set: The LDC for the Arabic Gigaword, the IXA NLP Research Group and Elhuyar Fundazioa for the Elhuyar Corpus Version, the French CNRTL for the Est Republicain Corpus, the IMS group and the university of Heidelberg for the German Wikidump, the Bar Ilan University for the Hebrew Wikidump, the University of Szeged for their Hungarian Newsire corpus, the KAIST for their Large Scale Korean Corpus, the Polish

Institute of Science for their Wikipedia corpus, and the Sprbanken group at the University of Gothenburg for their Parole corpus.

This work was supported by the National Science Foundation and the U.S. Army. Any opinion, content or information presented does not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

References

[Abeillé et al.2003] Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for french. In Anne Abeillé, editor, *Treebanks*. Kluwer, Dordrecht.

[Aduriz et al.2003] I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Díaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *TLT-03*, pages 201–204.

[Aldezabal et al.2008] I. Aldezabal, M.J. Aranzabe, A. Diaz de Ilarraza, and K. Fernández. 2008. From dependencies to constituents in the reference corpus for the processing of Basque. In *Procesamiento del Lenguaje Natural, n 41 (2008)*, pages 147–154. XXIV edición del Congreso Anual de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN).

[Björkelund et al.2013] Anders Björkelund, Ozlem Cetinoglu, Richárd Farkas, Thomas Mueller, and Wolfgang Seeker. 2013. (re)ranking meets morphosyntax: State-of-the-art results from the SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 135–145, Seattle, Washington, USA, October. Association for Computational Linguistics.

[Bohnet2010] Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China, August.

[Brants et al.2002] Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In Erhard Hinrichs and Kiril Simov, editors, *Proceedings of the First Workshop on Treebanks and Linguistic Theories (TLT 2002)*, pages 24–41, Sozopol, Bulgaria.

[Buchholz and Marsi2006] Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June. Association for Computational Linguistics.

[Candito et al.2010] Marie Candito, Benoit Crabbé, and Pascal Denis. 2010. Statistical French dependency parsing: Treebank conversion and first results. In *Proceedings of LREC*, Valletta, Malta.

[Choi et al.1994] Key-Sun Choi, Young S Han, Young G Han, and Oh W Kwon. 1994. Kaist tree bank project for korean: Present and future development. In *Proceedings of the International Workshop on Sharable Natural Language Resources*, pages 7–14. Citeseer.

[Choi2013] Jinho D Choi. 2013. Preparing korean data for the shared task on parsing morphologically rich languages. *arXiv preprint arXiv:1309.1649*.

[Collobert and Weston2008] R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning, ICML*.

[Csendes et al.2005] Dóra Csendes, János Csirik, Tibor Gyimóthy, and András Kocsor. 2005. The Szeged treebank. In *Proceedings of the 8th International Conference on Text, Speech and Dialogue (TSD)*, Lecture Notes in Computer Science, pages 123–132, Berlin / Heidelberg. Springer.

[Foth et al.2014] Kilian A. Foth, Arne Khn, Niels Beuck, and Wolfgang Menzel. 2014. Because size does matter: The hamburg dependency treebank. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may. European Language Resources Association (ELRA).

[Goldberg2011] Yoav Goldberg. 2011. *Automatic syntactic processing of Modern Hebrew*. Ph.D. thesis, Ben Gurion University of the Negev.

[Habash and Roth2009] Nizar Habash and Ryan Roth. 2009. Catib: The columbia arabic treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224, Suntec, Singapore, August. Association for Computational Linguistics.

[Habash et al.2009] Nizar Habash, Reem Faraj, and Ryan Roth. 2009. Syntactic Annotation in the Columbia Arabic Treebank. In *Proceedings of MEDAR International Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

[Lei et al.2014] Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1381–1391, Baltimore, Maryland, June. Association for Computational Linguistics.

[Maamouri et al.2004] Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a large-scale annotated arabic corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*.

[Martins et al.2013] Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 617–622, Sofia, Bulgaria, August.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

[Nivre et al.2006] Joakim Nivre, Jens Nilsson, and Johan Hall. 2006. Talbanken05: A Swedish treebank with phrase structure and dependency annotation. In *Proceedings of LREC*, pages 1392–1395, Genoa, Italy.

[Seddah et al.2013] Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, WA.

[Seeker and Kuhn2012] Wolfgang Seeker and Jonas Kuhn. 2012. Making Ellipses Explicit in Dependency Conversion for a German Treebank. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 3132–3139, Istanbul, Turkey. European Language Resources Association (ELRA).

[Sima'an et al.2001] Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman, and N. Nativ. 2001. Building a treebank of Modern Hebrew text. *Traitment Automatique des Langues*, 42(2).

[Świdziński and Woliński2010] Marek Świdziński and Marcin Woliński. 2010. Towards a bank of constituent parse trees for Polish. In *Proceedings of Text, Speech and Dialogue*, pages 197–204, Brno, Czech Republic.

[Tsarfaty2010] Reut Tsarfaty. 2010. Relational-realizational syntax: An architecture for specifying and statistically learning morphosyntactic descriptions. In *Proceedings of the LFG10 Conference*. CSLI Publications.

[Tsarfaty2013] Reut Tsarfaty. 2013. A unified morpho-syntactic scheme of stanford dependencies. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 578–584, Sofia, Bulgaria, August. Association for Computational Linguistics.

[Vincze et al.2010] Veronika Vincze, Dóra Szauter, Attila Almási, György Móra, Zoltán Alexin, and János Csirik. 2010. Hungarian Dependency Treebank. In *Proceedings of LREC*, Valletta, Malta.

[Woliński et al.2011] Marcin Woliński, Katarzyna Głowińska, and Marek Świdziński. 2011. A preliminary version of Składnica—a treebank of Polish. In *Proceedings of the 5th Language & Technology Conference*, pages 299–303, Poznań, Poland.

[Wróblewska2012] Alina Wróblewska. 2012. Polish Dependency Bank. *Linguistic Issues in Language Technology*, 7(1):1–15.