Data-driven Dependency Parsing With Empty Heads

Wolfgang Seeker *Richárd Farkas* *Bernd Bohnet*
Helmut Schmid *Jonas Kuhn*
Institut für Maschinelle Sprachverarbeitung
University of Stuttgart
firstname.lastname@ims.uni-stuttgart.de

ABSTRACT
Syntactic dependency structures are based on the assumption that there is exactly one node in the structure for each word in the sentence. However representing elliptical constructions (e.g. missing verbs) is problematic as the question where the dependents of the elided material should be attached to has to be solved. In this paper, we present an in-depth study into the challenges of introducing *empty heads* into dependency structures during automatic parsing. Structurally, empty heads provide an attachment site for the dependents of the non-overt material and thus preserve the linguistically plausible structure of the sentence. We compare three different (computational) approaches to the introduction of empty heads and evaluate them against German and Hungarian data. We then conduct a fine-grained error analysis on the output of one of the approaches to highlight some of the difficulties of the task. We find that while a clearly defined part of the phenomena can be learned by the parser, more involved elliptical structures are still mostly out of reach of the automatic tools.

KEYWORDS: Empty heads, statistical dependency parsing, German, Hungarian.

Proceedings of COLING 2012: Posters, pages 1081–1090,
COLING 2012, Mumbai, December 2012.

1 Introduction

Dependency structures (Mel'čuk, 1988) are a versatile formalism if one wants to represent syntactic structures for languages with free word order. Elliptical structures however can cause problems for this formalism because they violate a basic assumption, namely that there is exactly one node in the structure for each word in the sentence. In verbal ellipsis for example, this problem can break the entire structure because verbs often have dependents, but if the verb is not present in the sentence and subsequently not present in the structure, where should the dependents of the verb be attached to? One possibility that has been proposed is the introduction of *zero word forms* (Mel'čuk, 2009, 47), i. e. phonetically empty heads that appear in the structure but not on the surface string and provide an attachment site for the dependents of the ellipsis. In many languages, these constructions are rather frequent and should be handled by dependency parsers in a way that makes them easy to use in downstream applications.

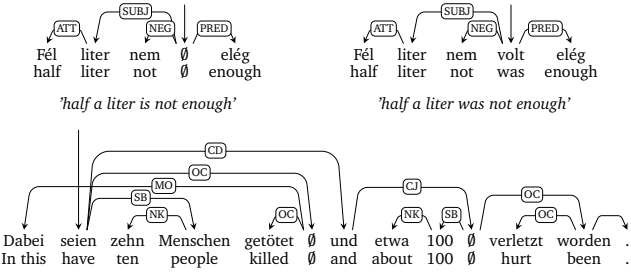

Figure 1: Empty heads in Hungarian (top) and German (bottom).

Figure 1 shows examples for two languages, Hungarian and German, where empty heads are annotated to ensure a linguistically plausible structure. In the top examples, a Hungarian copula construction is shown. In present tense (the sentence to the left), the copula is not overtly expressed and therefore represented as a phonetically empty head (Ø), while in all other tenses (right example), the copula would be expressed overtly. Since we would like the structure to be the same for both sentences, an empty head can be used to preserve the parallelism. The German example on the bottom shows a coordination of two sentences that share the finite and the passive auxiliary with each other, both represented as a phonetically empty head in the structure. By introducing empty nodes into the annotation, the parallelism in the underlying syntactic structure of the two conjuncts is preserved.

We would like to stress that the problem of empty heads in dependency syntax is rather different from the problem of introducing trace elements previously addressed by work on the English Penn Treebank (Johnson, 2002; Dienes and Dubey, 2003; Campbell, 2004). The PTB encodes a lot of different elements that do not show on the surface, but most of these would be leaf nodes in dependency representation.[1]

[1]There is a small number of cases, where the PTB annotates a missing verb (marked as *?*, see Section 4.6 in Bies et al. (1995)). We found 581 instances of those in the whole corpus, 293 of which were dominated by a VP node. Only those empty elements correspond to empty heads in a dependency representation since they would normally have dependents on their own. But in contrast to dependency formalisms, it is not a problem to annotate a head-less phrase in a phrase-structure formalism.

The aim of this paper is to investigate the problem of automatically predicting these empty heads in the process of statistical dependency parsing and to shed some light on the challenges of this problem. For this, we present and evaluate three different methods of introducing empty heads into dependency structures, one direct approach, where the parser itself decides when to annotate an empty head, one approach where the information about empty heads is encoded into the label set of the structure, and one approach where the presence of empty heads is determined by a classifier run prior to parsing.

The paper is structured as follows: we first review some related work and continue with the presentation of the three different methods. We then define the metric that we use to measure the quality of the empty head prediction and use it to evaluate parsing experiments on German and Hungarian. We conclude with an error analysis and a discussion of the results.

2 Related Work

We are aware of two previous papers where the issue of empty heads has been addressed in the context of dependency parsing: One is Dukes and Habash (2011) who present a parser for the Quranic Arabic Dependency Treebank, which also contains empty heads. Unfortunately, they do not evaluate or discuss the role of empty heads. Their solution of the problem – introducing a new transition into a transition-based parser – is similar to one of our proposed procedures (the in-parsing approach). The other work is by Chaitanya et al. (2011), who use hand-crafted rules to recover empty nodes from the output of a rule-based dependency parser for the Hindi Dependency Treebank. They achieve good results on some phenomena and a bit lower results on others, proving that it is indeed possible to treat this problem in syntactic processing. However, given that their data base is very small, and they used a rule-based, language-specific approach, the question remains if we can use statistical learning to address this problem.

3 Approaches for Parsing with Empty Heads

The parser that we use for our experiment is basically a best-first parser like the ones described in (Goldberg and Elhadad, 2010; Tratz and Hovy, 2011), which is trained with the Guided Learning technique proposed in Shen et al. (2007) embedded in a MIRA framework (Crammer et al., 2003). The best-first parsing approach has the advantage that it is easy to modify in order to allow for the introduction of empty heads, while for graph-based parsers (McDonald et al., 2005) it is not even clear how to do it. The approach is also more suitable here than the standard transition-based approach (Nivre et al., 2004), since it can build context on both sides of the current attachment site while it achieves competitive results.

In contrast to the best-first parser in Goldberg and Elhadad (2010), the decoding algorithm is modified so that it works like the LTAG dependency parser described in Shen and Joshi (2008), which allows an edge to attach to an inside node of an already built structure. This difference makes it possible to directly produce a large portion of non-projective structures (e. g. sentence extraposition or WH-extraction) without having to resort to a swap operation as is done in Tratz and Hovy (2011). It also increases theoretical decoding complexity to $O(n^2)$. However, there are non-projective structures that cannot be produced by this approach.[2] To allow the derivation of these structures, we reintroduce the swap operation from the parser in Tratz and Hovy (2011), but during training, the parser is only allowed to apply the swap operation in case of an ill-nested structure, which leads to a very small number of swaps.

[2]These structures do not fulfill the *well-nestedness condition* that is described in Bodirsky et al. (2005); Kuhlmann and Nivre (2006) and appear for example in German centerfield scrambling structures.

The **feature set** of the parser uses the word forms, lemmata, POS tags, and already predicted dependency labels for the head and its prospective dependent, as well as combinations thereof for up to three surrounding tokens in the sentence. The same features and combinations are extracted for up to three surrounding partially built structures. We also add features for the left-most and right-most dependent of a token, the labels of the dependents, distance features, and valency features as proposed by Zhang and Nivre (2011) but adapted to the best-first decoder. For internal feature representation and combination the parser implements the hash kernel method by Bohnet (2010).

3.1 Empty Head Introduction during Parsing

For the first method, we change the parser so that it can decide for an empty head during the parsing itself. To the three moves that the standard parser can perform – attach_left(label), attach_right(label), and swap – we add a fourth move (see Figure 2), that allows the parser to introduce an empty head for a particular dependent (together with a dependency label). This is similar in spirit to the parser presented in Dukes and Habash (2011), although they use a transition-based left to right approach for decoding. When training the parser, the empty heads in the training data are skipped in the feature extraction as long as they have not been predicted by the parser, and then added to the sentence as an additional node. If the parser makes a mistake during training, and the oracle is asked to provide the currently best-scoring valid action, we force it to defer proposing the introduction of an empty head as long as there is any other valid action available. This way we ensure that the decision to introduce an empty head is made only when the maximum syntactic context is available for the decision. During test time, we do not allow the parser to introduce two empty heads in a row to make sure that the parser does not enter an infinite loop of predicting empty heads over and over.

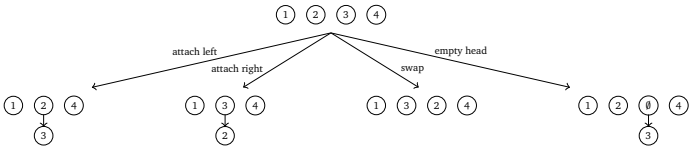

Figure 2: The four possible moves to extend the top configuration (unlabeled).

We add three basic features to the parser that are meant to capture some features of the empty heads. (1) we add a boolean feature that is true if there is a verb following the current token in the sentence. (2) we add a boolean feature that is true if there is at least one verb in the sentence. (3) we add a boolean feature that is true if there is at least one finite verb in the sentence. All three features are computed based on the POS tags of the tokens in the sentence. Finally, we also extract features (word form, pos, etc.) from the closest verb to the right and to the left of the current dependent. The boolean features are supposed to give information if there is any verb available in the sentence that could play the role of the main verb. Furthermore, since we assume for some of the elliptical constructions, that the verb that is missing is actually taking up another verb in the sentence, the latter features may capture this.

3.2 Standard Dependency Parsing with Complex Labels

In the second method, we encode information about empty heads in the label set of the treebank. Dependents of an empty head are attached to its head and their labels are combined with the

label of the empty head (see Figure 3). All the trees remain proper dependency trees where each node in the structure corresponds to a token in the input. The method is thus compatible with any standard parsing strategy. We apply the best-first parser to the complex-label data in order to gain a fair comparison with the previous approach. When the parser model with complex labels is applied to unseen data, it predicts the complex labels in the output, which then allow us to recover the empty heads. The recovery algorithm simply looks for all daughters of a node that are labeled with a complex label and introduces an empty node for each subset of nodes whose label prefixes match.[3]

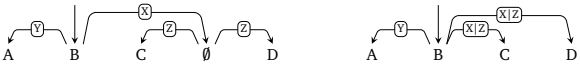

Figure 3: Encoding scheme for complex labels.

The complex label approach allows us to use any standard dependency parser. However, encoding information into labels needs to be done with caution. The label set can grow quickly, so the parsers will become much slower and will have a hard time learning the rare labels.

3.3 Preinserting Empty Heads

Besides the in-parsing approaches, we also experimented with a preinserting procedure as a third method. Here, the empty heads are automatically inserted into the sentence before parsing based on surface information. The advantage of this approach is again that we can use any standard dependency parser for parsing the input tokens and inserted empty word forms.

Our first attempt following the approach by (Dienes and Dubey, 2003) from the phrase-structure parsing literature shows very low accuracies. We attribute this to the different nature of the empty elements in the standard Penn Treebank setting and our data. The Penn Treebank traces can be modeled by local features and have fixed string positions (e. g. before to-infinitive constructions) whereas our empty heads can occur more freely due to the free word order of German and Hungarian.

We therefore pursue here a clause-based empty head preinsertion procedure since we think that the decision about inserting an empty head (which is basically the identification of the absence of the verb) can be made on the clause-level. For this, we implemented a clause boundary identification module and a classifier that predicts whether an empty word form should be inserted into a particular clause. Clause boundary identification is a difficult problem (cf. (Sang and Déjean, 2001)) as clauses usually form a hierarchy – and this hierarchy is important for predicting the insertion of empty heads. Our clause boundary detector achieves f-scores of 92.6 and 86.8 on the German and Hungarian development datasets respectively. These results are in line with the state-of-the-art results on the English Penn Treebank (Carreras et al., 2005; Ram and Lalitha Devi, 2008). If we evaluate only the in-sentence clauses, we get f-scores of 85.4 and 78.2 for German and Hungarian respectively.

In order to decide whether to insert an empty head, we implemented a classifier that decides for the insertion based on the output of the clause detector. The classifier utilizes the tokens, POS tags and morphological features of the clause and their patterns (bi- and trigrams) along with information about the hierarchy of the clauses (the depth of the clause and covered clauses).

[3]This is not a completely reversible procedure since we cannot recover two different empty nodes if they are sisters and happen to be labeled by the same label. However, this is a rare case in the treebanks (twice in the German treebank, five times in the Hungarian treebank).

The classifier achieves f-scores of 42.3 and 70.1 on the German and Hungarian development datasets respectively.

We use simple rules for finding the position inside the clause where the empty head should be inserted. For German, we insert it after the first noun sequence (which is an approximation of the place of the verb in the verb-second word order of German). For Hungarian, the manual annotation of the position of the empty word forms is quite irregular and we insert them at the beginning of the clause. Finally, we train the best-first parser on the original training dataset containing the gold standard empty heads and use it to parse the sentences that contain the automatically inserted empty heads from the preinserter.

4 Experiments

In order to test the parsing methods, we performed two experiments each: we trained the parser on the German TiGer corpus using the dependency representation by Seeker and Kuhn (2012) , and on the Szeged Dependency Treebank of Hungarian (Vincze et al., 2010), both of which data sets explicitly represent empty heads in the dependency trees. Table 1 shows the data sizes and the splits we used for the experiment. The German data was preprocessed (lemma, POS, morphology) with the mate-tools,[4] the Hungarian data comes with automatic annotation.[5]

		training		development		test	
data set	# sentences	# sents	# empty	# sents	# empty	# sents	# empty
German	50,474	36,000	2,618	2,000	117	10,472	722
Hungarian	81,960	61,034	14,850	11,688	2,536	9,238	2,106

Table 1: Data sets

4.1 Evaluation Method

Since the number of edges in the gold standard does not always equal the number of edges in the parser output if we allow the introduction of empty heads, we cannot use attachment accuracy anymore. To evaluate the recovery of empty heads, we therefore introduce three measures, which focus on different characteristics of the empty heads.[6]

The first metric (**ATTe**) is the labeled attachment score for empty heads only, where we count the correct attachment to the head as well as the correct attachment of the daughters of the empty head. A correct edge is thereby an edge that connects the same two tokens as in the gold standard, and that has the same label as the gold-standard edge, similar as in the ELAS score proposed by Dukes and Habash (2011), disregarding however the POS tag. One problem with the ELAS score is that it is not clear what happens if there is more than one empty head in the output of the parser or in the gold standard. In these cases, we compute the mapping of parser output empty heads to gold standard empty heads that maximizes the final score. This way, the evaluation is not skewed if the empty head is at the wrong string position, or if the number of empty heads in the gold standard differs from the number of empty heads in the parser output. As an extension to this metric, we define **ATTall**, which applies the ATTe to the whole node set in order to see the complete picture including the empty heads. For both metrics, we compute precision, recall, and f-score. In the last measure, **cLAS**, we use standard LAS by collapsing the

[4] http://code.google.com/p/mate-tools

[5] Our data sets and the evaluation tool will be made available on `www.ims.uni-stuttgart.de/~seeker`

[6] We do not evaluate string position since empty heads are only important for the structure.

empty heads in both parser output and gold standard using the complex label encoding. cLAS is a way of applying LAS to our parser output.

4.2 Experimental Results

The parser was trained on the training sets using 10 iterations and was then tested on the test sets. Table 2 presents the results in terms of the measures that we defined in Section 4.1.

		German				Hungarian			
		prec	rec	f1	acc	prec	rec	f1	acc
direct parsing	ATTe	56.81	24.46	34.20		69.08	63.07	65.94	
	ATTall	88.90	88.66	88.78		85.67	85.56	85.61	
	cLAS				88.90				85.28
complex labels	ATTe	58.82	28.77	38.64		67.56	59.36	63.20	
	ATTall	88.85	88.68	88.76		85.36	85.36	85.36	
	cLAS				88.90				85.18
preinsertion	ATTe	23.22	25.75	24.42		68.40	56.67	61.98	
	ATTall	88.09	88.11	88.10		85.25	85.02	85.14	
	cLAS				88.31				84.89

Table 2: Evaluation results on test sets.

Two general trends can be seen from the results. First, the preinsertion approach performs worse than the other two approaches. It is however not clear which of the other two is superior since for German, the complex label approach performs better while for Hungarian, the direct parsing approach comes out first. The second trend is that generally, the precision of the approach is much better than the recall, indicating that there are some phenomena that are relatively easy to learn resulting in a high precision, but most of the empty heads are not even recognized in the first place, hence the low recall. We also see that all approaches operate on a much higher level for Hungarian than for German.

5 Error Analysis

In this section, we perform a short error analysis to hint at some of the problems. We use the development sets of both data sets and the output of the direct parsing approach. Table 3 shows the performance on the development sets.

	German				Hungarian			
	prec	rec	f1	acc	prec	rec	f1	acc
ATTe	59.35	25.21	35.38		65.53	64.90	65.21	
ATTall	89.17	88.94	89.05		85.17	85.17	85.17	
cLAS				89.15				84.89

Table 3: Evaluation results on development sets.

One problem that we find when reviewing the output is that the parser often fails because of incorrect part-of-speech tagging. Sometimes, verbs are not recognized, which then prompts the parser to predict an empty head instead. Sometimes, other words are mistagged as verbs, which then prevents the parser from predicting an empty head. Table 4 shows the performance of the direct parsing approach when run on gold POS tags. The effect is greater for German than for Hungarian, but it is not substantial in both languages, showing that the parser relies strongly on POS information.

Another effect we find in the output is that some empty heads seem to be easier to predict than others. We illustrate this on the Hungarian data, where a coarse classification into empty copula (VAN) and other ellipsis (ELL) is annotated in the gold standard. Table 5 shows the

		German			Hungarian		
		prec	rec	f1	prec	rec	f1
predicted POS	ATTe	59.35	25.21	35.38	65.53	64.90	65.21
gold POS	ATTe	70.59	32.88	44.86	68.57	68.68	68.62

Table 4: Evaluation results on development sets using gold POS annotation.

performance on sentences that contain exactly one empty head in the gold standard.[7] The results indicate that empty copula are easier to learn for the parser than the more involved ellipses, which could be explained by the stronger grammaticalization of the former. Generally, the parser seems to rely heavily on lexical cues that do not vary much.

		ATTe		
	#sentence	prec	rec	f1
VAN	1760	83.64	69.23	75.76
ELL	307	80.18	42.56	55.61

Table 5: Results achieved on two different types of empty heads (Hungarian).

	prec	rec	f1	acc
ATTe	95.37	66.16	78.13	
ATTall	98.88	98.73	98.81	
cLAS				98.85

Table 6: Applying the direct parsing approach to its own training set (German).

Finally, we show the performance of the direct parsing approach when applied to its own training set in German. This can be instructive to see if the features of the statistical model actually capture the correct information to identify empty heads. The results (Table 6) suggest that they do not. A standard statistical dependency parser (without empty heads) usually achieves scores in the high 90s on its own training data. Results clearly show that work in the feature set of the parser is necessary to predict empty heads more accurately, but also that the standard syntactic features are not capable of modeling the phenomenon.

Conclusion and Future Work

Empty heads represent material that is missing on the surface of the sentence but is understood and usually easily reconstructible by humans. Formally, empty heads provide attachment sites for their overtly expressed dependents and thus help representing syntactic structures of elliptic constructions in the same way as their non-elliptic counterparts. In this paper, we evaluated three methods of introducing empty heads into a dependency structure, a direct approach during parsing, an encoding scheme that allows the reconstruction of empty heads after standard parsing, and a preinsertion approach where the empty heads are predicted prior to parsing. All three methods were evaluated on German and Hungarian.

The preinsertion method is outperformed by the two other approaches, but which of these is superior remains to be seen. In general, no method performs on a satisfactory level indicating that empty heads are a difficult phenomenon. Our error analysis shows that standard features for statistical dependency parsing are not able to model empty heads convincingly. One thing that we can learn from our investigation is, that it is probably wise to separate the empty heads that we have in the data into at least two parts, namely the part that contains constructions like the Hungarian copula (and probably similar local, more grammaticized constructions), and the other more involved ellipses. The parser seems to be able to predict missing copula reasonably well and this can be used to predict more accessible output structures. Since the construction also appears in other languages, e. g. the Slavic languages, this may turn out useful.

[7] We only choose these sentences, because the type of a predicted empty head is not straight forward to determine automatically.

Acknowledgments

We would like to thank Bernadette Rauschenberger for her help. This work was funded by the Deutsche Forschungsgemeinschaft (DFG) via Projects D4 and D8 of the SFB 732 "Incremental Specification in Context".

References

Bies, A., Ferguson, M., Katz, K., and MacIntyre, R. (1995). Bracketing Guidelines for Treebank II style Penn Treebank Project. Technical report, Linguistic Data Consortium.

Bodirsky, M., Kuhlmann, M., and Möhl, M. (2005). Well-nested drawings as models of syntactic structure. In *Proceedings of the 10th Conference on Formal Grammar and 9th Meeting on Mathematics of Language*, pages 195–204.

Bohnet, B. (2010). Very high 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. International Committee on Computational Linguistics.

Campbell, R. (2004). Using linguistic principles to recover empty categories. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics - ACL 2004*, pages 645–es, Morristown, NJ, USA. Association for Computational Linguistics.

Carreras, A. X., Màrquez, B. L., and Castro, C. J. (2005). Filtering-ranking perceptron learning for partial parsing. *Machine Learning*, 60:41–71. 10.1007/s10994-005-0917-x.

Chaitanya, G., Husain, S., and Mannem, P. (2011). Empty categories in Hindi dependency treebank: analysis and recovery. In *Proceedings of the 5th Linguistic Annotation Workshop (LAW 2011)*, Portland, USA. Association for Computational Linguistics.

Crammer, K., Dekel, O., Shalev-Shwartz, S., and Singer, Y. (2003). Online passive-aggressive algorithms. In *Proceedings of the 16th Annual Conference on Neural Information Processing Systems (NIPS)*, volume 7. MIT Press.

Dienes, P. and Dubey, A. (2003). Deep syntactic processing by combining shallow methods. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, volume 1, pages 431–438, Sapporo, Japan. Association for Computational Linguistics.

Dukes, K. and Habash, N. (2011). One-Step Statistical Parsing of Hybrid Dependency-Constituency Syntactic Representations. In *Proceedings of the International Conference on Parsing Technology (IWPT 2011)*, pages 92–104, Dublin, Ireland. Association for Computational Linguistics.

Goldberg, Y. and Elhadad, M. (2010). An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2010)*, pages 742–750. Association for Computational Linguistics.

Johnson, M. (2002). A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, page 136, Morristown, NJ, USA. Association for Computational Linguistics.

Kuhlmann, M. and Nivre, J. (2006). Mildly non-projective dependency structures. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 507–514, Morristown, NJ, USA. Association for Computational Linguistics.

McDonald, R., Crammer, K., and Pereira, F. (2005). Online large-margin training of dependency parsers. In *Proceedings of ACL 2005*, pages 91–98, Morristown, NJ, USA. Association for Computational Linguistics.

Mel'čuk, I. (1988). *Dependency Syntax: Theory and Practice*. State University Press of New York, suny serie edition.

Mel'čuk, I. (2009). *Dependency in linguistic description*.

Nivre, J., Hall, J., and Nilsson, J. (2004). Memory-based dependency parsing. In *Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL 2004)*, pages 49–56, Boston, Massachusetts.

Ram, R. and Lalitha Devi, S. (2008). Clause boundary identification using conditional random fields. In Gelbukh, A., editor, *Computational Linguistics and Intelligent Text Processing*, volume 4919 of *Lecture Notes in Computer Science*, pages 140–150. Springer Berlin Heidelberg.

Sang, E. F. T. K. and Déjean, H. (2001). Introduction to the conll-2001 shared task: clause identification. In *Proceedings of the ACL 2001 Workshop on Computational Natural Language Learning (ConLL)*.

Seeker, W. and Kuhn, J. (2012). Making Ellipses Explicit in Dependency Conversion for a German Treebank. In *Proceedings of the 8th Language Resources and Evaluation Conference (LREC 2012)*, pages 3132–3139, Istanbul, Turkey. European Language Resources Association (ELRA).

Shen, L. and Joshi, A. K. (2008). Ltag dependency parsing with bidirectional incremental construction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 495–504, Honolulu, Hawaii. Association for Computational Linguistics.

Shen, L., Satta, G., and Joshi, A. K. (2007). Guided Learning for Bidirectional Sequence Classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*, pages 760–767. Association for Computational Linguistics.

Tratz, S. and Hovy, E. (2011). A Fast, Accurate, Non-Projective, Semantically-Enriched Parser. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pages 1257–1268, Edinburgh, UK.

Vincze, V., Szauter, D., Almási, A., Móra, G., Alexin, Z., and Csirik, J. (2010). Hungarian Dependency Treebank. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC 2010)*, pages 1855–1862, Valletta, Malta.

Zhang, Y. and Nivre, J. (2011). Transition-based Dependency Parsing with Rich Non-local Features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers*, pages 188–193, Portland, USA. Association for Computational Linguistics.