On WordNet Semantic Classes and Dependency Parsing

Kepa Bengoetxea†, Eneko Agirre†, Joakim Nivre‡,
Yue Zhang*, Koldo Gojenola†
†University of the Basque Country UPV/EHU / IXA NLP Group
‡Uppsala University / Department of Linguistics and Philology
∗ Singapore University of Technology and Design
kepa.bengoetxea@ehu.es, e.agirre@ehu.es,
joakim.nivre@lingfil.uu.se, yue_zhang@sutd.edu.sg,
koldo.gojenola@ehu.es

Abstract

This paper presents experiments with WordNet semantic classes to improve dependency parsing. We study the effect of semantic classes in three dependency parsers, using two types of constituency-to-dependency conversions of the English Penn Treebank. Overall, we can say that the improvements are small and not significant using automatic POS tags, contrary to previously published results using gold POS tags (Agirre et al., 2011). In addition, we explore parser combinations, showing that the semantically enhanced parsers yield a small significant gain only on the more semantically oriented LTH treebank conversion.

1 Introduction

This work presents a set of experiments to investigate the use of lexical semantic information in dependency parsing of English. Whether semantics improve parsing is one interesting research topic both on parsing and lexical semantics. Broadly speaking, we can classify the methods to incorporate semantic information into parsers in two: systems using static lexical semantic repositories, such as WordNet or similar ontologies (Agirre et al., 2008; Agirre et al., 2011; Fujita et al., 2010), and systems using dynamic semantic clusters automatically acquired from corpora (Koo et al., 2008; Suzuki et al., 2009).

Our main objective will be to determine whether static semantic knowledge can help parsing. We will apply different types of semantic information to three dependency parsers. Specifically, we will test the following questions:

- Does semantic information in WordNet help dependency parsing? Agirre et al. (2011) found improvements in dependency parsing using MaltParser on gold POS tags. In this work, we will investigate the effect of semantic information using predicted POS tags.
- Is the type of semantic information related to the type of parser? We will test three different parsers representative of successful paradigms in dependency parsing.
- How does the semantic information relate to the style of dependency annotation? Most experiments for English were evaluated on the Penn2Malt conversion of the constituency-based Penn Treebank. We will also examine the LTH conversion, with richer structure and an extended set of dependency labels.
- How does WordNet compare to automatically obtained information? For the sake of comparison, we will also perform the experiments using syntactic/semantic clusters automatically acquired from corpora.
- Does parser combination benefit from semantic information? Different parsers can use semantic information in diverse ways. For example, while MaltParser can use the semantic information in local contexts, MST can incorporate them in global contexts. We will run parser combination experiments with and without semantic information, to determine whether it is useful in the combined parsers.

After introducing related work in section 2, section 3 describes the treebank conversions, parsers and semantic features. Section 4 presents the results and section 5 draws the main conclusions.

2 Related work

Broadly speaking, we can classify the attempts to add external knowledge to a parser in two sets: using large semantic repositories such as WordNet and approaches that use information automatically acquired from corpora. In the first group, Agirre et al. (2008) trained two state-of-the-art constituency-based statistical parsers (Charniak,

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 649–655,
Baltimore, Maryland, USA, June 23-25 2014.

2000; Bikel, 2004) on semantically-enriched input, substituting content words with their semantic classes, trying to overcome the limitations of lexicalized approaches to parsing (Collins, 2003) where related words, like *scissors* and *knife*, cannot be generalized. The results showed a significant improvement, giving the first results over both WordNet and the Penn Treebank (PTB) to show that semantics helps parsing. Later, Agirre et al. (2011) successfully introduced WordNet classes in a dependency parser, obtaining improvements on the full PTB using gold POS tags, trying different combinations of semantic classes. MacKinlay et al. (2012) investigate the addition of semantic annotations in the form of word sense hypernyms, in HPSG parse ranking, reducing error rate in dependency F-score by 1%, while some methods produce substantial decreases in performance. Fujita et al. (2010) showed that fully disambiguated sense-based features smoothed using ontological information are effective for parse selection.

On the second group, Koo et al. (2008) presented a semisupervised method for training dependency parsers, introducing features that incorporate word clusters automatically acquired from a large unannotated corpus. The clusters include strongly semantic associations like {apple, pear} or {Apple, IBM} and also syntactic clusters like {of, in}. They demonstrated its effectiveness in dependency parsing experiments on the PTB and the Prague Dependency Treebank. Suzuki et al. (2009), Sagae and Gordon (2009) and Candito and Seddah (2010) also experiment with the same cluster method. Recently, Täckström et al. (2012) tested the incorporation of cluster features from unlabeled corpora in a multilingual setting, giving an algorithm for inducing cross-lingual clusters.

3 Experimental Framework

In this section we will briefly describe the PTB-based datasets (subsection 3.1), followed by the data-driven parsers used for the experiments (subsection 3.2). Finally, we will describe the different types of semantic representation that were used.

3.1 Treebank conversions

Penn2Malt[1] performs a simple and direct conversion from the constituency-based PTB to a dependency treebank. It obtains projective trees and has been used in several works, which allows us to compare our results with related experiments (Koo et al., 2008; Suzuki et al., 2009; Koo and Collins, 2010). We extracted dependencies using standard head rules (Yamada and Matsumoto, 2003), and a reduced set of 12 general dependency tags.

LTH[2] (Johansson and Nugues, 2007) presents a conversion better suited for semantic processing, with a richer structure and a more fine-grained set of dependency labels (42 different dependency labels), including links to handle long-distance phenomena, giving a 6.17% of nonprojective sentences. The results from parsing the LTH output are lower than those for Penn2Malt conversions.

3.2 Parsers

We have made use of three parsers representative of successful paradigms in dependency parsing.

MaltParser (Nivre et al., 2007) is a deterministic transition-based dependency parser that obtains a dependency tree in linear-time in a single pass over the input using a stack of partially analyzed items and the remaining input sequence, by means of history-based feature models. We added two features that inspect the semantic feature at the top of the stack and the next input token.

MST[3] represents global, exhaustive graph-based parsing (McDonald et al., 2005; McDonald et al., 2006) that finds the highest scoring directed spanning tree in a graph. The learning procedure is global since model parameters are set relative to classifying the entire dependency graph, in contrast to the local but richer contexts used by transition-based parsers. The system can be trained using first or second order models. The second order projective algorithm performed best on both conversions, and we used it in the rest of the evaluations. We modified the system in order to add semantic features, combining them with wordforms and POS tags, on the parent and child nodes of each arc.

ZPar[4] (Zhang and Clark, 2008; Zhang and Nivre, 2011) performs transition-based dependency parsing with a stack of partial analysis and a queue of remaining inputs. In contrast to MaltParser (local model and greedy deterministic search) ZPar applies global discriminative learning and beam search. We extend the feature set of ZPar to include semantic features. Each set of semantic information is represented by two atomic

[1] http://w3.msi.vxu.se/ nivre/research/Penn2Malt.html

[2] http://nlp.cs.lth.se/software/treebank_converter

[3] http://mstparser.sourceforge.net

[4] www.sourceforge.net/projects/zpar

	Base line	WordNet SF	WordNet SS	Clusters
Malt	88.46	88.49 (+0.03)	88.42 (-0.04)	88.59 (+0.13)
MST	90.55	90.70 (+0.15)	90.47 (-0.08)	90.88 (+0.33)‡
ZPar	91.52	91.65 (+0.13)	91.70 (+0.18)†	91.74 (+0.22)

Table 1: LAS results with several parsing algorithms, Penn2Malt conversion (†: p <0.05, ‡: p <0.005). In parenthesis, difference with baseline.

	Base line	WordNet SF	WordNet SS	Clusters
Malt	84.95	85.12 (+0.17)	85.08 (+0.16)	85.13 (+0.18)
MST	85.06	85.35 (+0.29)‡	84.99 (-0.07)	86.18 (+1.12)‡
ZPar	89.15	89.33 (+0.18)	89.19 (+0.04)	89.17 (+0.02)

Table 2: LAS results with several parsing algorithms in the LTH conversion (†: p <0.05, ‡: p <0.005). In parenthesis, difference with baseline.

feature templates, associated with the top of the stack and the head of the queue, respectively. ZPar was directly trained on the Penn2Malt conversion, while we applied the pseudo-projective transformation (Nilsson et al., 2008) on LTH, in order to deal with non-projective arcs.

3.3 Semantic information

Our aim was to experiment with different types of WordNet-related semantic information. For comparison with automatically acquired information, we will also experiment with bit clusters.

WordNet. We will experiment with the semantic representations used in Agirre et al. (2008) and Agirre et al. (2011), based on WordNet 2.1. WordNet is organized into sets of synonyms, called synsets (SS). Each synset in turn belongs to a unique semantic file (SF). There are a total of 45 SFs (1 for adverbs, 3 for adjectives, 15 for verbs, and 26 for nouns), based on syntactic and semantic categories. For example, noun SFs differentiate nouns denoting acts or actions, and nouns denoting animals, among others. We experiment with both full SSs and SFs as instances of fine-grained and coarse-grained semantic representation, respectively. As an example, *knife* in its tool sense is in the EDGE TOOL USED AS A CUTTING INSTRUMENT singleton synset, and also in the ARTIFACT SF along with thousands of words including *cutter*. These are the two extremes of semantic granularity in WordNet. For each semantic representation, we need to determine the semantics of each occurrence of a target word. Agirre et al. (2011) used i) gold-standard annotations from SemCor, a subset of the PTB, to give an upper bound performance of the semantic representation, ii) first sense, where all instances of a word were tagged with their most frequent sense, and iii) automatic sense ranking, predicting the most frequent sense for each word (McCarthy et al., 2004). As we will make use of the full PTB, we only have access to the first sense information.

Clusters. Koo et al. (2008) describe a semi-supervised approach that makes use of cluster features induced from unlabeled data, providing significant performance improvements for supervised dependency parsers on the Penn Treebank for English and the Prague Dependency Treebank for Czech. The process defines a hierarchical clustering of the words, which can be represented as a binary tree where each node is associated to a bit-string, from the more general (root of the tree) to the more specific (leaves). Using prefixes of various lengths, it can produce clusterings of different granularities. It can be seen as a representation of syntactic-semantic information acquired from corpora. They use short strings of 4-6 bits to represent parts of speech and the full strings for wordforms.

4 Results

In all the experiments we employed a baseline feature set using word forms and parts of speech, and an enriched feature set (WordNet or clusters). We firstly tested the addition of each individual semantic feature to each parser, evaluating its contribution to the parser's performance. For the combinations, instead of feature-engineering each parser with the wide array of different possibilities for features, as in Agirre et al. (2011), we adopted the simpler approach of combining the outputs of the individual parsers by voting (Sagae and Lavie, 2006). We will use Labeled Attachment Score (LAS) as our main evaluation criteria. As in previous work, we exclude punctuation marks. For all the tests, we used a perceptron POS-tagger (Collins, 2002), trained on WSJ sections 2–21, to assign POS tags automatically to both the training (using 10-way jackknifing) and test data, obtaining a POS tagging accuracy of 97.32% on the test data. We will make use of Bikel's randomized parsing evaluation comparator to test the statistical significance of the results. In all of the experiments the parsers were trained on sections 2-21 of the PTB and evaluated on the development set (section 22). Finally, the best performing system was evaluated on the test set (section 23).

Parsers	LAS	UAS
Best baseline (ZPar)	91.52	92.57
Best single parser (ZPar + Clusters)	91.74 (+0.22)	92.63
Best combination (3 baseline parsers)	91.90 (+0.38)	93.01
Best combination of 3 parsers: 3 baselines + 3 SF extensions	91.93 (+0.41)	92.95
Best combination of 3 parsers: 3 baselines + 3 SS extensions	91.87 (+0.35)	92.92
Best combination of 3 parsers: 3 baselines + 3 cluster extensions	91.90 (+0.38)	92.90

Table 3: Parser combinations on Penn2Malt.

Parsers	LAS	UAS
Best baseline (ZPar)	89.15	91.81
Best single parser (ZPar + SF)	89.33 (+0.15)	92.01
Best combination (3 baseline parsers)	89.15 (+0.00)	91.81
Best combination of 3 parsers: 3 baselines + 3 SF extensions	89.56 (+0.41)‡	92.23
Best combination of 3 parsers: 3 baselines + 3 SS extensions	89.43 (+0.28)	93.12
Best combination of 3 parsers: 3 baselines + 3 cluster extensions	89.52 (+0.37)†	92.19

Table 4: Parser combinations on LTH (†: $p < 0.05$, ‡: $p < 0.005$).

4.1 Single Parsers

We run a series of experiments testing each individual semantic feature, also trying different learning configurations for each one. Regarding the WordNet information, there were 2 different features to experiment with (SF and SS). For the bit clusters, there are different possibilities, depending on the number of bits used. For Malt and MST, all the different lengths of bit strings were used. Given the computational requirements and the previous results on Malt and MST, we only tested all bits in ZPar. Tables 1 and 2 show the results.

Penn2Malt. Table 1 shows that the only significant increase over the baseline is for ZPar with SS and for MST with clusters.

LTH. Looking at table 2, we can say that the differences in baseline parser performance are accentuated when using the LTH treebank conversion, as ZPar clearly outperforms the other two parsers by more than 4 absolute points. We can see that SF helps all parsers, although it is only significant for MST. Bit clusters improve significantly MST, with the highest increase across the table.

Overall, we see that the small improvements do not confirm the previous results on Penn2Malt, MaltParser and gold POS tags. We can also conclude that automatically acquired clusters are specially effective with the MST parser in both treebank conversions, which suggests that the type of semantic information has a direct relation to the parsing algorithm. Section 4.3 will look at the details by each knowledge type.

4.2 Combinations

Subsection 4.1 presented the results of the base algorithms and their extensions based on semantic features. Sagae and Lavie (2006) report improvements over the best single parser when combining three transition-based models and one graph-based model. The same technique was also used by the winning team of the CoNLL 2007 Shared Task (Hall et al., 2007), combining six transition-based parsers. We used MaltBlender[5], a tool for merging the output of several dependency parsers, using the Chu-Liu/Edmonds directed MST algorithm. After several tests we noticed that weighted voting by each parser's labeled accuracy gave good results, using it in the rest of the experiments. We trained different types of combination:

- Base algorithms. This set includes the 3 baseline algorithms, MaltParser, MST, and ZPar.
- Extended parsers, adding semantic information to the baselines. We include the three base algorithms and their semantic extensions (SF, SS, and clusters). It is known (Surdeanu and Manning, 2010) that adding more parsers to an ensemble usually improves accuracy, as long as they add to the diversity (and almost regardless of their accuracy level). So, for the comparison to be fair, we will compare ensembles of 3 parsers, taken from sets of 6 parsers (3 baselines + 3 SF, SS, and cluster extensions, respectively).

In each experiment, we took the best combination of individual parsers on the development set for the final test. Tables 3 and 4 show the results.

Penn2Malt. Table 3 shows that the combination of the baselines, without any semantic information, considerably improves the best baseline. Adding semantics does not give a noticeable increase with respect to combining the baselines.

LTH (table 4). Combining the 3 baselines does not give an improvement over the best baseline, as ZPar clearly outperforms the other parsers. However, adding the semantic parsers gives an increase with respect to the best single parser (ZPar + SF), which is small but significant for SF and clusters.

4.3 Analysis

In this section we analyze the data trying to understand where and how semantic information helps most. One of the obstacles of automatic parsers is the presence of incorrect POS tags due to auto-

[5] http://w3.msi.vxu.se/users/jni/blend/

POS tags	Parser	LAS test set	LAS on sentences without POS errors	LAS on sentences with POS errors
Gold	ZPar	90.45	91.68	89.14
Automatic	ZPar	89.15	91.62	86.51
Automatic	Best combination of 3 parsers: 3 baselines + 3 SF extensions	89.56 (+0.41)	91.90 (+0.28)	87.06 (+0.55)
Automatic	Best combination of 3 parsers: 3 baselines + 3 SS extensions	89.43 (+0.28)	91.95 (+0.33)	86.75 (+0.24)
Automatic	Best combination of 3 parsers: 3 baselines + 3 cluster extensions	89.52 (+0.37)	91.92 (+0.30)	86.96 (+0.45)

Table 5: Differences in LAS (LTH) for baseline and extended parsers with sentences having correct/incorrect POS tags (the parentheses show the difference w.r.t ZPar with automatic POS tags).

matic tagging. For example, ZPar's LAS score on the LTH conversion drops from 90.45% with gold POS tags to 89.12% with automatic POS tags. We will examine the influence of each type of semantic information on sentences that contain or not POS errors, and this will clarify whether the increments obtained when using semantic information are useful for correcting the negative influence of POS errors or they are orthogonal and constitute a source of new information independent of POS tags. With this objective in mind, we analyzed the performance on the subset of the test corpus containing the sentences which had POS errors (1,025 sentences and 27,300 tokens) and the subset where the sentences had (automatically assigned) correct POS tags (1,391 sentences and 29,386 tokens).

Table 5 presents the results of the best single parser on the LTH conversion (ZPar) with gold and automatic POS tags in the first two rows. The LAS scores are particularized for sentences that contain or not POS errors. The following three rows present the enhanced (combined) parsers that make use of semantic information. As the combination of the three baseline parsers did not give any improvement over the best single parser (ZPar), we can hypothesize that the gain coming from the parser combinations comes mostly from the addition of semantic information. Table 5 suggests that the improvements coming from WordNet's semantic file (SF) are unevenly distributed between the sentences that contain POS errors and those that do not (an increase of 0.28 for sentences without POS errors and 0.55 for those with errors). This could mean that a big part of the information contained in SF helps to alleviate the errors performed by the automatic POS tagger. On the other hand, the increments are more evenly distributed for SS and clusters, and this can be due to the fact that the semantic information is orthogonal to the POS, giving similar improvements for sentences that contain or not POS errors. We independently tested this fact for the individual parsers. For example, with MST and SF the gains almost doubled for sentences with incorrect POS tags (+0.37 with respect to +0.21 for sentences with correct POS tags) while the gains of adding clusters' information for sentences without and with POS errors were similar (0.91 and 1.33, repectively). This aspect deserves further investigation, as the improvements seem to be related to both the type of semantic information and the parsing algorithm.We did an initial exploration but it did not give any clear indication of the types of improvements that could be expected using each parser and semantic data.

5 Conclusions

This work has tried to shed light on the contribution of semantic information to dependency parsing. The experiments were thorough, testing two treebank conversions and three parsing paradigms on automatically predicted POS tags. Compared to (Agirre et al., 2011), which used MaltParser on the LTH conversion and gold POS tags, our results can be seen as a negative outcome, as the improvements are very small and non-significant in most of the cases. For parser combination, WordNet semantic file information does give a small significant increment in the more fine-grained LTH representation. In addition we show that the improvement of automatic clusters is also weak. For the future, we think tdifferent parsers, eitherhat a more elaborate scheme is needed for word classes, requiring to explore different levels of generalization in the WordNet (or alternative) hierarchies.

Acknowledgments

This research was supported by the the Basque Government (IT344- 10, S PE11UN114), the University of the Basque Country (GIU09/19) and the Spanish Ministry of Science and Innovation (MICINN, TIN2010-20218).

References

Eneko Agirre, Timothy Baldwin, and David Martinez. 2008. Improving parsing and PP attachment performance with sense information. In *Proceedings of ACL-08: HLT*, pages 317–325, Columbus, Ohio, June. Association for Computational Linguistics.

Eneko Agirre, Kepa Bengoetxea, Koldo Gojenola, and Joakim Nivre. 2011. Improving dependency parsing with semantic classes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 699–703, Portland, Oregon, USA, June. Association for Computational Linguistics.

Daniel M. Bikel. 2004. Intricacies of collins' parsing model. *Computational Linguistics*, 30(4):479–511.

Marie Candito and Djamé Seddah. 2010. Parsing word clusters. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 76–84, Los Angeles, CA, USA, June. Association for Computational Linguistics.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, NAACL 2000, pages 132–139, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637, December.

Sanae Fujita, Francis Bond, Stephan Oepen, and Takaaki Tanaka. 2010. Exploiting semantic information for hpsg parse selection. *Research on Language and Computation*, 8(1):122.

Johan Hall, Jens Nilsson, Joakim Nivre, Glsen Eryigit, Beta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? a study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task EMNLP-CoNLL*.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, pages 105–112, Tartu, Estonia, May 25-26.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, Uppsala, Sweden, July. Association for Computational Linguistics.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio, June. Association for Computational Linguistics.

Andrew MacKinlay, Rebecca Dridan, Diana McCarthy, and Timothy Baldwin. 2012. The effects of semantic annotations on precision parse ranking. In *First Joint Conference on Lexical and Computational Semantics (*SEM)*, page 228236, Montreal, Canada, June. Association for Computational Linguistics.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 279–286, Barcelona, Spain, July.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of CoNLL 2006*.

Jens Nilsson, Joakim Nivre, and Johan Hall. 2008. Generalizing tree transformations for inductive dependency parsing. In *Proceedings of the 45th Conference of the ACL*.

Joakim Nivre, Johan Hall, Jens Nilsson, Chanev A., Glsen Eryiit, Sandra Kbler, Marinov S., and Edwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*.

Kenji Sagae and Andrew Gordon. 2009. Clustering words by syntactic similarity improves dependency parsing of predicate-argument structures. In *Proceedings of the Eleventh International Conference on Parsing Technologies*.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Proceedings of the North American Chapter of the Association for Computational Linguistics Conference (NAACL-2010)*, Los Angeles, CA, June.

Jun Suzuki, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 551–560, Singapore, August. Association for Computational Linguistics.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 477–487, Montréal, Canada, June. Association for Computational Linguistics.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th IWPT*, pages 195–206. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571, Honolulu, Hawaii, October. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 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*, pages 188–193, Portland, Oregon, USA, June. Association for Computational Linguistics.