Dependency Parsing with Short Dependency Relations in Unlabeled Data

Wenliang Chen, Daisuke Kawahara, Kiyotaka Uchimoto, Yujie Zhang, Hitoshi Isahara
Computational Linguistics Group
National Institute of Information and Communications Technology
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan, 619-0289
{chenwl, dk, uchimoto, yujie, isahara}@nict.go.jp

Abstract

This paper presents an effective dependency parsing approach of incorporating short dependency information from unlabeled data. The unlabeled data is automatically parsed by a deterministic dependency parser, which can provide relatively high performance for short dependencies between words. We then train another parser which uses the information on short dependency relations extracted from the output of the first parser. Our proposed approach achieves an unlabeled attachment score of 86.52, an absolute 1.24% improvement over the baseline system on the data set of Chinese Treebank.

1 Introduction

In dependency parsing, we attempt to build the dependency links between words from a sentence. Given sufficient labeled data, there are several supervised learning methods for training high-performance dependency parsers(Nivre et al., 2007). However, current statistical dependency parsers provide worse results if the dependency length becomes longer (McDonald and Nivre, 2007). Here the length of a dependency from word w_i and word w_j is simply equal to $|i - j|$. Figure 1 shows the F_1 score[1] provided by a deterministic parser relative to dependency length on our testing data. From the figure, we find that F_1 score decreases when dependency length increases as (McDonald and Nivre, 2007) found. We also notice that the parser provides good results for short dependencies (94.57% for dependency length = 1 and 89.40% for dependency length = 2). In this paper, short dependency refers to the dependencies whose length is 1 or 2.

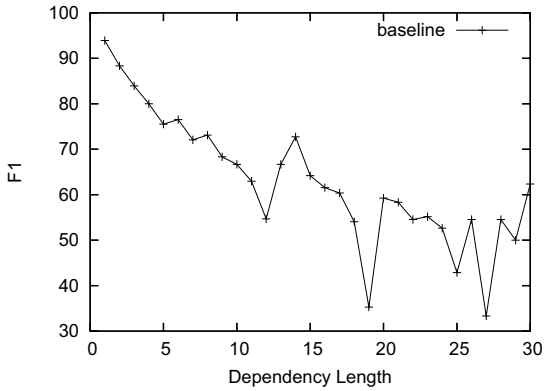

Figure 1: F-score relative to dependency length

Labeled data is expensive, while unlabeled data can be obtained easily. In this paper, we present an approach of incorporating unlabeled data for dependency parsing. First, all the sentences in unlabeled data are parsed by a dependency parser, which can provide state-of-the-art performance. We then extract information on short dependency relations from the parsed data, because the performance for short dependencies is relatively higher than others. Finally, we train another parser by using the information as features.

The proposed method can be regarded as a semi-supervised learning method. Currently, most semi-

[1] precision represents the percentage of predicted arcs of length d that are correct and recall measures the percentage of gold standard arcs of length d that are correctly predicted.
$F_1 = 2 \times precision \times recall/(precision + recall)$

supervised methods seem to do well with artificially restricted labeled data, but they are unable to outperform the best supervised baseline when more labeled data is added. In our experiments, we show that our approach significantly outperforms a state-of-the-art parser, which is trained on full labeled data.

2 Motivation and previous work

The goal in dependency parsing is to tag dependency links that show the head-modifier relations between words. A simple example is in Figure 2, where the link between *a* and *bird* denotes that *a* is the dependent of the head *bird*.

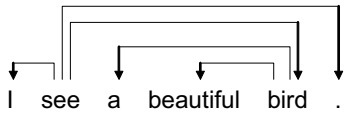

Figure 2: Example dependency graph.

We define that ***word distance*** of word w_i and word w_j is equal to $|i-j|$. Usually, the two words in a head-dependent relation in one sentence can be adjacent words (word distance = 1) or neighboring words (word distance = 2) in other sentences. For example, "a" and "bird" has head-dependent relation in the sentence at Figure 2. They can also be adjacent words in the sentence "I see a bird.".

Suppose that our task is Chinese dependency parsing. Here, the string "专家级JJ(Specialist-level)/工　作NN(working)/会　谈NN(discussion)" should be tagged as the solution (a) in Figure 3. However, our current parser may choose the solution (b) in Figure 3 without any additional information. The point is how to assign the head for "专家级(Specialist-level)". Is it "工作(working)" or "会谈(discussion)"?

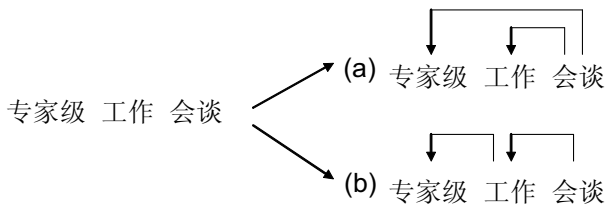

Figure 3: Two solutions for "专家级(Specialist-level)/工作(working)/会谈(discussion)"

As Figure 1 suggests, the current dependency parser is good at tagging the relation between adjacent words. Thus, we expect that dependencies of adjacent words can provide useful information for parsing words, whose word distances are longer. When we search the string "专家级(Specialist-level)/会谈(discussion)" at google.com, many relevant documents can be retrieved. If we have a good parser, we may assign the relations between the two words in the retrieved documents as Figure 4 shows. We can find that "会谈(discussion)" is the head of "专家级(Specialist-level)" in many cases.

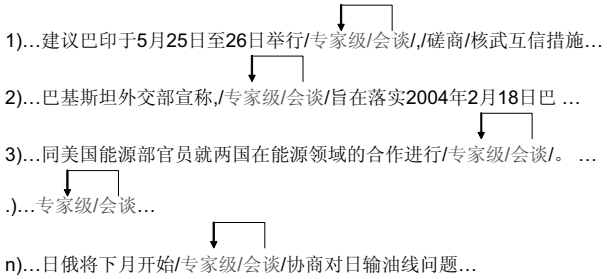

Figure 4: Parsing "专家级(Specialist-level)/会谈(discussion)" in unlabeled data

Now, consider what a learning model could do to assign the appropriate relation between "专家级(Specialist-level)" and "会谈(discussion)" in the string "专家级(Specialist-level)/工作(working)/会谈(discussion)". In this case, we provide additional information to "会谈(discussion)" as the possible head of "专家级(Specialist-level)" in the unlabeled data. In this way, the learning model may use this information to make correct decision.

Till now, we demonstrate how to use the dependency relation between adjacent words in unlabeled data to help tag the relation between two words whose word distance is 2. In the similar way, we can also assign the relation between two words whose word distance is longer by using the information.

Based on the above observations, we propose an approach of exploiting the information from a large-scale unlabeled data for dependency parsing. We use a parser to parse the sentences in unlabeled data. Then another parser makes use of the information on short dependency relations in the newly parsed data to improve performance.

Our study is relative to incorporating unlabeled

data into a model for parsing. There are several other studies relevant to ours as described below.

A simple method is self-training in which the existing model first labels unlabeled data and then the newly labeled data is then treated as hand annotated data for training a new model. But it seems that self-training is not so effective. (Steedman et al., 2003) reports minor improvement by using self-training for syntactic parsing on small labeled data. The reason may be that errors in the original model would be amplified in the new model. (McClosky et al., 2006) presents a successful instance of parsing with self-training by using a re-ranker. As Figure 1 suggests, the dependency parser performs bad for parsing the words with long distances. In our approach, we choose partial reliable information which comes from short dependency relations for the dependency parser.

(Smith and Eisner, 2006) presents an approach to improve the accuracy of a dependency grammar induction models by EM from unlabeled data. They obtain consistent improvements by penalizing dependencies between two words that are farther apart in the string.

The study most relevant to ours is done by (Kawahara and Kurohashi, 2006). They present an integrated probabilistic model for Japanese parsing. They also use partial information after current parser parses the sentences. Our work differs in that we consider general dependency relations while they only consider case frames. And we represent additional information as the features for learning models while they use the case frames as one component for a probabilistic model.

3 Our Approach

In this section, we describe our approach of exploiting reliable features from unlabeled data, which is parsed by a basic parser. We then train another parser based on new feature space.

3.1 Training a basic parser

In this paper, we implement a deterministic parser based on the model described by (Nivre, 2003). This model is simple and works very well in the shared-tasks of CoNLL2006(Nivre et al., 2006) and CoNLL2007(Hall et al., 2007). In fact, our approach can also be applied to other parsers, such as (Yamada and Matsumoto, 2003)'s parser, (McDonald et al., 2006)'s parser, and so on.

3.1.1 The parser

The parser predicts unlabeled directed dependencies between words in sentences. The algorithm (Nivre, 2003) makes a dependency parsing tree in one left-to-right pass over the input, and uses a stack to store the processed tokens. The behaviors of the parser are defined by four elementary actions (where TOP is the token on top of the stack and NEXT is the next token in the original input string):

- Left-Arc(LA): Add an arc from NEXT to TOP; pop the stack.
- Right-Arc(RA): Add an arc from TOP to NEXT; push NEXT onto the stack.
- Reduce(RE): Pop the stack.
- Shift(SH): Push NEXT onto the stack.

The first two actions mean that there is a dependency relation between TOP and NEXT.

More information about the parser can be available in the paper(Nivre, 2003). The parser uses a classifier to produce a sequence of actions for a sentence. In our experiments, we use the SVM model as the classifier. More specifically, our parser uses LIBSVM(Chang and Lin, 2001) with a polynomial kernel (degree = 3) and the built-in one-versus-all strategy for multi-class classification.

3.1.2 Basic features

We represent basic features extracted from the fields of data representation, including word and part-of-speech(POS) tags. The basic features used in our parser are listed as follows:

- The features based on words: the words of TOP and NEXT, the word of the head of TOP, the words of leftmost and rightmost dependent of TOP, and the word of the token immediately after NEXT in original input string.
- The features based on POS: the POS of TOP and NEXT, the POS of the token immediately below TOP, the POS of leftmost and rightmost dependent of TOP, the POS of next three tokens after NEXT, and the POS of the token immediately before NEXT in original input string.

With these basic features, we can train a state-of-the-art supervised parser on labeled data. In the following content, we call this parser Basic Parser.

3.2 Unlabeled data preprocessing and parsing

The input of our approach is unlabeled data, which can be obtained easily. For the Basic Parser, the corpus should have part-of-speech (POS) tags. Therefore, we should assign the POS tags using a POS tagger. For Chinese sentences, we should segment the sentences into words before POS tagging. After data preprocessing, we have the word-segmented sentences with POS tags. We then use the Basic Parser to parse all sentences in unlabeled data.

3.3 Using short dependency relations as features

The Basic Parser can provide complete dependency parsing trees for all sentences in unlabeled data. As Figure 1 shows, short dependencies are more reliable. To offer reliable information for the model, we propose the features based on short dependency relations from the newly parsed data.

3.3.1 Collecting reliable information

In a parsed sentence, if the dependency length of two words is 1 or 2, we add this word pair into a list named DepList and count its frequency. We consider the direction and length of the dependency. D1 refers to the pairs with dependency length 1, D2 refers to the pairs with dependency length 2, R refers to right arc, and L refers to left arc. For example, "专家级(specialist-level)" and "会谈(discussion)" are adjacent words in a sentence "我们(We)/举行(held)/专家级(specialist-level)/会谈(discussion)/。" and have a left dependency arc assigned by the Basic Parser. We add a word pair "专家级(specialist-level)-会谈(discussion)" with "D1-L" and its frequency into the DepList.

According to frequency, we then group word pairs into different buckets, with a bucket ONE for frequency 1, a single bucket LOW for 2-7, a single bucket MID for 8-14, and a single bucket HIGH for 15+. We choose these threshold values via testing on development data. For example, the frequency of the pair "专家级(specialist-level)-会谈(discussion)" with "D1-L" is 20. Then it is grouped into the bucket "D1-L-HIGH".

Here, we do not use the frequencies as the weight of the features. We derive the weights of the features by the SVM model from training data rather than approximating the weights from unlabeled data.

3.3.2 New features

Based on the DepList, we represent new features for training or parsing current two words: TOP and NEXT. We consider word pairs from the context around TOP and NEXT, and get the buckets of the pairs in the DepList.

First, we represent the features based on D1. We name these features D1 features. The D1 features are listed according to different word distances between TOP and NEXT as follows:

1. Word distance is 1: (TN0) the bucket of the word pair of TOP and NEXT, and (TN1) the bucket of the word pair of TOP and next token after NEXT.
2. Word distance is 2 or 3+: (TN0) the bucket of the word pair of TOP and NEXT, (TN1) the bucket of the word pair of TOP and next token after NEXT, and (TN_1) the bucket of the word pair of TOP and the token immediately before NEXT.

In item 2), all features are in turn combined with two sets of distances: a set for distance 2 and a single set for distances 3+. Thus, we have 8 types of D1 features, including 2 types in item 1) and 6 types in item 2). The feature is formatted as "Position:WordDistance:PairBucket". For example, we have the string "专家级(specialist-level)/w_1/w_2/w_3/会谈(discussion)", and "专家级(specialist-level)" is TOP and "会谈(discussion)" is NEXT. Thus we can have the feature "TN0:3+:D1-L-HIGH" for TOP and NEXT, because the word distance is 4(3+) and "专家级(specialist-level)-会谈(discussion)" belongs to the bucket "D1-L-HIGH". Here, if a string belongs to two buckets, we use the most frequent bucket.

Then, we represent the features based on D2. We name these features D2 features. The D2 features are listed as follows:

1. Word distance is 1: (TN1) the bucket of the word pair of TOP and next token after NEXT.

2. Word distance is 2: (TN0) the bucket of the word pair of TOP and NEXT, and (TN1) the bucket of the word pair of TOP and next token after NEXT.

4 Experiments

For labeled data, we used the Chinese Treebank (CTB) version 4.0[2] in our experiments. We used the same rules for conversion and created the same data split as (Wang et al., 2007): files 1-270 and 400-931 as training, 271-300 as testing and files 301-325 as development. We used the gold standard segmentation and POS tags in the CTB.

For unlabeled data, we used the PFR corpus [3]. It includes the documents from People's Daily at 1998 (12 months). There are about 290 thousand sentences and 15 million words in the PFR corpus. To simplify, we used its segmentation. And we discarded the POS tags because PFR and CTB used different POS sets. We used the package TNT (Brants, 2000), a very efficient statistical part-of-speech tagger, to train a POS tagger[4] on training data of the CTB.

We measured the quality of the parser by the unlabeled attachment score (UAS), i.e., the percentage of tokens with correct HEAD. We reported two types of scores: "UAS without p" is the UAS score without all punctuation tokens and "UAS with p" is the one with all punctuation tokens.

4.1 Experimental results

In the experiments, we trained the parsers on training data and tuned the parameters on development data. In the following sessions, "baseline" refers to Basic Parser (the model with basic features), and "OURS" refers to our proposed parser (the model with all features).

4.1.1 Our approach

Table 1 shows the results of the parser with different feature sets, where "+D1" refers to the parser with basic features and D1 features, and "+D2" refers to the parser with all features(basic features, D1 features, and D2 features). From the table, we found a large improvement (1.12% for UAS without p and 1.23% for UAS with p) from adding D1 features. And D2 features provided minor improvement, 0.12% for UAS without p and 0.14% for UAS with p. This may be due to the information from dependency length 2 containing more noise. Totally, we achieved 1.24% improvement for UAS with p and 1.37% for UAS without p. The improvement is significant in one-tail paired t-test ($p < 10^{-5}$).

Table 1: The results with different feature sets

	UAS without p	UAS with p
baseline	85.28	83.79
+D1	86.40	85.02
+D2(OURS)	86.52	85.16

We also attempted to discover the effect of different numbers of unlabeled sentences to use. Table 2 shows the results with different numbers of sentences. Here, we randomly chose different percentages of sentences from unlabeled data. When we used 1% sentences of unlabeled data, the parser achieved a large improvement. As we added more sentences, the parser obtained more benefit.

Table 2: The results with different numbers of unlabeled sentences

Sentences	UAS without p	UAS with p
0%(baseline)	85.28	83.79
1%	85.68	84.40
2%	85.69	84.51
5%	85.78	84.59
10%	85.97	84.62
20%	86.25	84.86
50%	86.34	84.92
100%(OURS)	86.52	85.16

4.1.2 Comparison of other systems

Finally, we compare our parser to the state of the art. We used the same testing data as (Wang et al., 2005) did, selecting the sentences length up to 40. Table 3 shows the results achieved by our method and other researchers (UAS with p), where Wang05 refers to (Wang et al., 2005), Wang07 refers

[2]More detailed information can be found at http://www.cis.upenn.edu/~chinese/.

[3]More detailed information can be found at http://www.icl.pku.edu.

[4]To know whether our POS tagger is good, we also tested the TNT package on the standard training and testing sets for full parsing (Wang et al., 2006). The TNT-based tagger provided 91.52% accuracy, the comparative result with (Wang et al., 2006).

to (Wang et al., 2007), and McDonald&Pereira06[5] refers to (McDonald and Pereira, 2006). From the table, we found that our parser performed best.

Table 3: The results on the sentences length up to 40

	UAS with p
Wang05	79.9
McDonald&Pereira06	82.5
Wang07	86.6
baseline	87.1
OURS	88.4

5 Analysis

5.1 Improvement relative to dependency length

We now look at the improvement relative to dependency length as Figure 5 shows. From the figure, we found that our method provided better performance when dependency lengths are less than 13. Especially, we had improvements 2.35% for dependency length 4, 3.13% for length 5, 2.56% for length 6, and 4.90% for length 7. For longer ones, the parser can not provide stable improvement. The reason may be that shorter dependencies are often modifier of nouns such as determiners or adjectives or pronouns modifying their direct neighbors, while longer dependencies typically represent modifiers of the root or the main verb in a sentence(McDonald and Nivre, 2007). We did not provide new features for modifiers of the root.

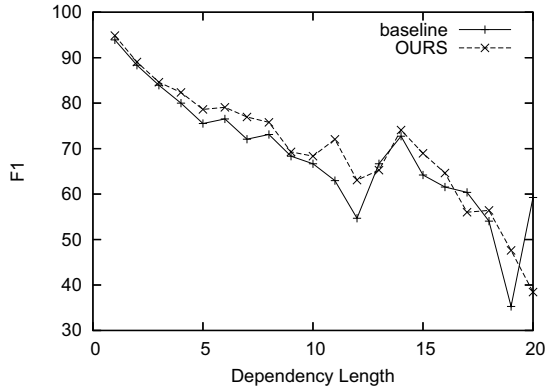

Figure 5: Improvement relative to dependency length

[5](Wang, 2007) reported this result.

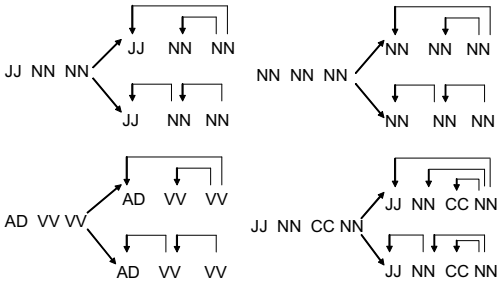

Figure 6: Ambiguities

5.2 Cases study in neighborhood

In Chinese dependency parsing, there are many ambiguities in neighborhood, such as "JJ NN NN", "AD VV VV", "NN NN NN", "JJ NN CC NN". They have possible parsing trees as Figure 6 shows. For these ambiguities, our approach can provide additional information for the parser. For example, we have the following case in the data set: "友好JJ(friendly)/ 合作NN(corporation)/ 关系NN(relationship)/". We can provide additional information about the relations of "友好JJ(friendly)/ 合作NN(corporation)" and "友好JJ(friendly)/ 关系NN(relationship)/" in unlabeled data to help the parser make the correct decision.

Our approach can also work for the longer constructions, such as "JJ NN NN NN" and "NN NN NN NN" in the similar way.

For the construction "JJ NN1 CC NN2", we now do not define special features to solve the ambiguity. However, based on the current DepList, we can also provide additional information about the relations of JJ/NN1 and JJ/NN2. For example, for the string "进一步JJ(further)/ 改善NN(improvement)/ 和CC(and)/ 发展NN(development)/", the parser often assigns "改善(improvement)" as the head of "进一步(further)" instead of "发展(development)". There is an entry "进一步(further)-发展(development)" in the DepList. Here, we need a coordination identifier to identify these constructions. After that, we can provide the information for the model.

6 Conclusion

This paper presents an effective approach to improve dependency parsing by using unlabeled data. We extract the information on short dependency relations

in an automatically generated corpus parsed by a basic parser. We then train a new parser with the information. The new parser achieves an absolute improvement of 1.24% over the state-of-the-art parser on Chinese Treebank (from 85.28% to 86.52%).

There are many ways in which this research should be continued. First, feature representation needs to be improved. Here, we use a simple feature representation on short dependency relations. We may use a combined representation to use the information from long dependency relations even they are not so reliable. Second, we can try to select more accurately parsed sentences. Then we may collect more reliable information than the current one.

References

T. Brants. 2000. TnT–a statistical part-of-speech tagger. *Proceedings of the 6th Conference on Applied Natural Language Processing*, pages 224–231.

C.C. Chang and C.J. Lin. 2001. LIBSVM: a library for support vector machines. *Software available at http://www. csie. ntu. edu. tw/cjlin/libsvm*, 80:604–611.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryigit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? a study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 933–939.

D. Kawahara and S. Kurohashi. 2006. A fully-lexicalized probabilistic model for Japanese syntactic and case structure analysis. *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 176–183.

D. McClosky, E. Charniak, and M. Johnson. 2006. Reranking and self-training for parser adaptation. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 337–344.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of the 11th Conf. of the European Chapter of the ACL (EACL)*.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 216–220, New York City, June. Association for Computational Linguistics.

J. Nivre, J. Hall, J. Nilsson, G. Eryigit, and S Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

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

Noah A. Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 569–576, Sydney, Australia, July. Association for Computational Linguistics.

M. Steedman, M. Osborne, A. Sarkar, S. Clark, R. Hwa, J. Hockenmaier, P. Ruhlen, S. Baker, and J. Crim. 2003. Bootstrapping statistical parsers from small datasets. *The Proceedings of the Annual Meeting of the European Chapter of the ACL*, pages 331–338.

Qin Iris Wang, Dale Schuurmans, and Dekang Lin. 2005. Strictly lexical dependency parsing. In *IWPT2005*.

Mengqiu Wang, Kenji Sagae, and Teruko Mitamura. 2006. A Fast, Accurate Deterministic Parser for Chinese. In *Coling-ACL2006*.

Qin Iris Wang, Dekang Lin, and Dale Schuurmans. 2007. Simple training of dependency parsers via structured boosting. In *IJCAI2007*.

Qin Iris Wang. 2007. Learning structured classifiers for statistical dependency parsing. In *NAACL-HLT 2007 Doctoral Consortium*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of the 8th Intern. Workshop on Parsing Technologies (IWPT)*, pages 195–206.