A Two-stage Parser for Multilingual Dependency Parsing

Wenliang Chen, 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, yujie, isahara}@nict.go.jp

Abstract

We present a two-stage multilingual dependency parsing system submitted to the Multilingual Track of CoNLL-2007. The parser first identifies dependencies using a deterministic parsing method and then labels those dependencies as a sequence labeling problem. We describe the features used in each stage. For four languages with different values of ROOT, we design some special features for the ROOT labeler. Then we present evaluation results and error analyses focusing on Chinese.

1 Introduction

The CoNLL-2007 shared tasks include two tracks: the Multilingual Track and Domain Adaptation Track(Nivre et al., 2007). We took part the Multilingual Track of all ten languages provided by the CoNLL-2007 shared task organizers(Hajič et al., 2004; Aduriz et al., 2003; Martí et al., 2007; Chen et al., 2003; Böhmová et al., 2003; Marcus et al., 1993; Johansson and Nugues, 2007; Prokopidis et al., 2005; Csendes et al., 2005; Montemagni et al., 2003; Oflazer et al., 2003) .

In this paper, we describe a two-stage parsing system consisting of an unlabeled parser and a sequence labeler, which was submitted to the Multilingual Track. At the first stage, we use the parsing model proposed by (Nivre, 2003) to assign the arcs between the words. Then we obtain a dependency parsing tree based on the arcs. At the second stage, we use a SVM-based approach(Kudo and Matsumoto, 2001) to tag the dependency label for each arc. The labeling is treated as a sequence labeling problem. We design some special features for tagging the labels of ROOT for Arabic, Basque, Czech, and Greek, which have different labels for ROOT. The experimental results show that our approach can provide higher scores than average.

2 Two-Stage Parsing

2.1 The Unlabeled Parser

The unlabeled parser predicts unlabeled directed dependencies. This parser is primarily based on the parsing models described by (Nivre, 2003). The algorithm 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.

Although (Nivre et al., 2006) used the pseudo-projective approach to process non-projective dependencies, here we only derive projective dependency tree. We use MaltParser(Nivre et al., 2006)

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1129–1133,
Prague, June 2007.

V0.4[1] to implement the unlabeled parser, and use the SVM model as the classifier. More specifically, the MaltParser use LIBSVM(Chang and Lin, 2001) with a quadratic kernel and the built-in one-versus-all strategy for multi-class classification.

2.1.1 Features for Parsing

The MaltParser is a history-based parsing model, which relies on features of the derivation history to predict the next parser action. We represent the features extracted from the fields of the data representation, including FORM, LEMMA, CPOSTAG, POSTAG, and FEATS. We use the features for all languages that are listed as follows:

- The FORM features: the FORM of TOP and NEXT, the FORM of the token immediately before NEXT in original input string, and the FORM of the head of TOP.
- The LEMMA features: the LEMMA of TOP and NEXT, the LEMMA of the token immediately before NEXT in original input string, and the LEMMA of the head of TOP.
- The CPOS features: the CPOSTAG of TOP and NEXT, and the CPOSTAG of next left token of the head of TOP.
- The POS features: the POSTAG of TOP and NEXT, the POSTAG of next three tokens after NEXT, the POSTAG of the token immediately before NEXT in original input string, the POSTAG of the token immediately below TOP, and the POSTAG of the token immediately after rightmost dependent of TOP.
- The FEATS features: the FEATS of TOP and NEXT.

But note that the fields LEMMA and FEATS are not available for all languages.

2.2 The Sequence Labeler

2.2.1 The Sequence Problem

We denote by $x = x_1, ..., x_n$ a sentence with n words and by y a corresponding dependency tree. A dependency tree is represented from ROOT to leaves with a set of ordered pairs $(i, j) \in y$ in which x_j is a dependent and x_i is the head. We have produced the dependency tree y at the first stage. In this stage, we assign a label $l_{(i,j)}$ to each pair.

As described in (McDonald et al., 2006), we treat the labeling of dependencies as a sequence labeling problem. Suppose that we consider a head x_i with dependents $x_{j1}, ..., x_{jM}$. We then consider the labels of $(i, j1), ..., (i, jM)$ as a sequence. We use the model to find the solution:

$$l_{max} = \arg\max_{l} s(l, i, y, x) \tag{1}$$

And we consider a first-order Markov chain of labels.

We used the package YamCha (V0.33)[2] to implement the SVM model for labeling. YamCha is a powerful tool for sequence labeling(Kudo and Matsumoto, 2001).

2.2.2 Features for Labeling

After the first stage, we know the unlabeled dependency parsing tree for the input sentence. This information forms the basis for part of the features of the second stage. For the sequence labeler, we define the individual features, the pair features, the verb features, the neighbor features, and the position features. All the features are listed as follows:

- The individual features: the FORM, the LEMMA, the CPOSTAG, the POSTAG, and the FEATS of the parent and child node.
- The pair features: the direction of dependency, the combination of lemmata of the parent and child node, the combination of parent's LEMMA and child's CPOSTAG, the combination of parent's CPOSTAG and child's LEMMA, and the combination of FEATS of parent and child.
- The verb features: whether the parent or child is the first or last verb in the sentence.
- The neighbor features: the combination of CPOSTAG and LEMMA of the left and right neighbors of the parent and child, number of children, CPOSTAG sequence of children.

[1] The tool is available at http://w3.msi.vxu.se/˜nivre/research/MaltParser.html

[2] YamCha is available at http://chasen.org/˜taku/software/yamcha/

- The position features: whether the child is the first or last word in the sentence and whether the child is the first word of left or right of parent.

2.2.3 Features for the Root Labeler

Because there are four languages have different labels for root, we define the features for the root labeler. The features are listed as follows:

- The individual features: the FORM, the LEMMA, the CPOSTAG, the POSTAG, and the FEATS of the parent and child node.

- The verb features: whether the child is the first or last verb in the sentence.

- The neighbor features: the combination of CPOSTAG and LEMMA of the left and right neighbors of the parent and child, number of children, CPOSTAG sequence of children.

- The position features: whether the child is the first or last word in the sentence and whether the child is the first word of left or right of parent.

3 Evaluation Results

We evaluated our system in the Multilingual Track for all languages. For the unlabeled parser, we chose the parameters for the MaltParser based on performance from a held-out section of the training data. We also chose the parameters for Yamcha based on performance from training data.

Our official results are shown at Table 1. Performance is measured by labeled accuracy and unlabeled accuracy. These results showed that our two-stage system can achieve good performance. For all languages, our system provided better results than average performance of all the systems(Nivre et al., 2007). Compared with top 3 scores, our system provided slightly worse performance. The reasons may be that we just used projective parsing algorithms while all languages except Chinese have non-projective structure. Another reason was that we did not tune good parameters for the system due to lack of time.

Data Set	LA	UA
Arabic	74.65	83.49
Basque	72.39	78.63
Catalan	86.66	90.87
Chinese	81.24	85.91
Czech	73.69	80.14
English	83.81	84.91
Greek	74.42	81.16
Hungarian	75.34	79.25
Italian	82.04	85.91
Turkish	76.31	81.92
average	78.06	83.22

Table 1: The results of proposed approach. LABELED ATTACHMENT SCORE(LA) and UNLABELED ATTACHMENT SCORE(UA)

4 General Error Analysis

4.1 Chinese

For Chinese, the system achieved 81.24% on labeled accuracy and 85.91% on unlabeled accuracy. We also ran the MaltParser to provide the labels. Besides the same features, we added the DEPREL features: the dependency type of TOP, the dependency type of the token leftmost of TOP, the dependency type of the token rightmost of TOP, and the dependency type of the token leftmost of NEXT. The labeled accuracy of MaltParser was 80.84%, 0.4% lower than our system.

Some conjunctions, prepositions, and DE[3] attached to their head words with much lower accuracy: 74% for DE, 76% for conjunctions, and 71% for prepositions. In the test data, these words formed 19.7%. For Chinese parsing, coordination and preposition phrase attachment were hard problems. (Chen et al., 2006) defined the special features for coordinations for chunking. In the future, we plan to define some special features for these words.

Now we focused words where most of the errors occur as Table 2 shows. For “的/DE”, there was 32.4% error rate of 383 occurrences. And most of them were assigned incorrect labels between “property” and “predication”: 45 times for “property” instead of “predication” and 20 times for “predication” instead of “property”. For examples, “的/DE”

[3] including “的/得/地/之”.

	num	any	head	dep	both
的/ DE	383	124	35	116	27
、/ C	117	38	36	37	35
在/ P	67	20	6	19	5
台灣/ N	31	10	8	4	2
是/ V	72	8	8	8	8

Table 2: The words where most of errors occur in Chinese data.

in "流行/的/電視/頻道(popular TV channel)" was to be tagged as "property" instead of "predication", while "的/DE" in "博物館/的/志工(volunteer of museum)" was to be tagged as "predication" instead of "property". It was very hard to tell the labels between the words around "的". Humans can make the distinction between property and predication for "的", because we have background knowledge of the words. So if we can incorporate the additional knowledge for the system, the system may assign the correct label.

For "、/C", it was hard to assign the head, 36 wrong head of all 38 errors. It often appeared at coordination expressions. For example, the head of "、" at "在/酷/斃/了/、/太/炫/了/之外/(Besides extreme cool and too amazing)" was "之外", and the head of "、" at "提供給/參觀/者/深厚/、/有/系統/的/知識(Give the visitors solid and methodical knowledge)" was "知識".

5 Conclusion

In this paper, we presented our two-stage dependency parsing system submitted to the Multilingual Track of CoNLL-2007 shared task. We used Nivre's method to produce the dependency arcs and the sequence labeler to produce the dependency labels. The experimental results showed that our system can provide good performance for all languages.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7, pages 103–127.

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.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006. An empirical study of chinese chunking. In *COLING/ACL 2006(Poster Sessions)*, Sydney, Australia, July.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *In Proceedings of NAACL01*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

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.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (Abeillé, 2003), chapter 11, pages 189–210.

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.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.