Improving Dependency Parsers with Supertags

Hiroki Ouchi **Kevin Duh** **Yuji Matsumoto**
Computational Linguistics Laboratory
Nara Institute of Science and Technology
{ouchi.hiroki.nt6, kevinduh, matsu}@is.naist.jp

Abstract

Transition-based dependency parsing systems can utilize rich feature representations. However, in practice, features are generally limited to combinations of lexical tokens and part-of-speech tags. In this paper, we investigate richer features based on supertags, which represent lexical templates extracted from dependency structure annotated corpus. First, we develop two types of supertags that encode information about head position and dependency relations in different levels of granularity. Then, we propose a transition-based dependency parser that incorporates the predictions from a CRF-based supertagger as new features. On standard English Penn Treebank corpus, we show that our supertag features achieve parsing improvements of 1.3% in unlabeled attachment, 2.07% root attachment, and 3.94% in complete tree accuracy.

1 Introduction

One significant advantage of transition-based dependency parsing (Yamada and Matsumoto, 2003; Nivre et al, 2007, Goldberg and Elhadad, 2010; Huang and Sagae, 2010) is that they can utilize rich feature representations. However, in practice, current state-of-the-art parsers generally utilize only features that are based on lexical tokens and part-of-speech (POS) tags. In this paper, we argue that more complex features that capture fine-grained syntactic phenomenon and long-distance dependencies represent a simple and effective way to improve transition-based dependency parsers.

We focus on defining supertags for English dependency parsing. Supertags, which are lexical templates extracted from dependency structure annotated corpus, encode linguistically rich information that imposes complex constraints in a local context (Bangalore and Joshi, 1999). While supertags have been used in frameworks based on lexicalized grammars, e.g. Lexicalized Tree-Adjoining Grammar (LTAG), Head-driven Phrase Structure Grammar (HPSG) and Combinatory Categorial Grammar (CCG), they have scarcely been utilized for dependency parsing so far.

Previous work by Foth et al (2006) demonstrate that supertags improve German dependency parsing under a Weighted Constraint Dependency Grammar (WCDG). Recent work by Ambati et al (2013) show that supertags based on CCG lexicon improves transition-based dependency parsing for Hindi. In particular, they argue that supertags can improve long distance dependencies (e.g. coordination, relative clause) in a morphologically-rich free-word-order language. Zhang et. al. (2010) define supertags that incorporate that long-distance dependency information for the purpose of HPSG parsing. All these works suggest the promising synergy between dependency parsing and supertagging. Our main contributions are: (1) an investigation of supertags that work well for English dependency parsing, and (2) a novel transition-based parser that effectively utilizes such supertag features.

In the following, we first describe our supertag design (Section 2) and parser (Section 3). Supertagging and parsing experiments on the Penn Treebank (Marcus et al., 1993) are shown in Section 4. We show that using automatically predicted supertags, our parser can achieve improvements of 1.3% in unlabeled attachment, 2.07% root attachment, and 3.94% in complete tree accuracy.

2 Supertag Design

The main challenge with designing supertags is finding the right balance between granularity and predictability. Ideally, we would like to increase the granularity of the supertags in order capture

Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 154–158,
Gothenburg, Sweden, April 26-30 2014.

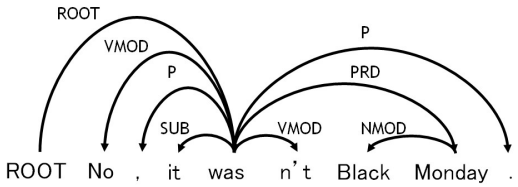

Figure 1: Example sentence

Word	Model 1	Model 2
No	VMOD/R	VMOD/R
,	P/R	P/R
it	SUB/R	SUB/R
was	ROOT+L_R	ROOT+SUB/L_PRD/R
n't	VMOD/L	VMOD/L
Black	NMOD/R	NMOD/R
Monday	PRD/L+L	PRD/L+L
.	P/L	P/L

Table 1: Model 1 & 2 supertags for Fig. 1.

more fine-grained syntactic information, but large tagsets tend to be more difficult to predict automatically. We describe two supertag designs with different levels of granularity in the following, focusing on incorporating syntactic features that we believe are important for dependency parsing.

For easy exposition, consider the example sentence in Figure 1. Our first supertag design, Model 1, represents syntactic information that shows the relative position (direction) of the head of a word, such as left (L) or right (R). If a word has root as its head, we consider it as no direction. In addition, dependency relation labels of heads are added. For instance, 'No' in the example in Figure 1 has its head in the right direction with a label 'VMOD', so its supertag can be represented as 'VMOD/R'. This kind of information essentially provides clues about the role of the word in sentence.

On top of this, we also add information about whether a word has any left or right dependents. For instance, the word 'Monday' has a left dependent 'Black', so we encode it as 'PRD/L+L', where the part before '+' specifies the head information ('PRD/L') and the part afterwards ('L') specifies the position of the dependent ('L' for left, 'R' for right). When a word has its dependents in both left and right directions, such as the word 'was' in Figure 1, we combine them using '_', as in: 'ROOT+L_R'. On our Penn Treebank data, Model 1 has 79 supertags.

unigrams of supertags	
for p in p_{i-2}, p_{i-1}, p_i, p_{i+1}, p_{i+2}, p_{i+3}	$w_p s_p$, $t_p s_p$
bigrams of supertags	
for p, q in (p_i, p_{i+1}), (p_i, p_{i+2}), (p_{i-1}, p_i), (p_{i-1}, p_{i+2}), (p_{i+1}, p_{i+2})	$s_p s_q$, $t_p s_q$, $s_p t_q$, $w_p s_q$, $s_p w_q$
head-dependent of supertags	
for p, q in (p_i, p_{i+1}), (p_i, p_{i+2}), (p_{i-1}, p_i), (p_{i-1}, p_{i+2}), (p_{i+1}, p_{i+2})	$w_p sh_p w_q sld_q$, $t_p sh_p t_q sld_q$, $w_p srd_p w_q sh_q$, $t_p srd_p t_q sh_q$

Table 2: Proposed supertag feature templates. w = word; t = POS-tag; s = supertag; sh = head part of supertag; sld = left dependent part of supertag; srd = right dependent part of supertag

In Model 2, we further add dependency relation labels of obligatory dependents of verbs. Here we define obligatory dependents of verbs as dependents which have the following dependency relation labels, 'SUB', 'OBJ', 'PRD' and 'VC'. If a label of a dependent is not any of the obligatory dependent labels, the supertag encodes only the information of direction of the dependents (same as Model 1). For instance, 'was' in the example sentence has an obligatory dependent with a label 'SUB' in the left direction and 'PRD' in the right direction, so its supertag is represented as 'ROOT+SUB/L_PRD/R'. If a verb has multiple obligatory dependents in the same direction, its supertag encodes them in sequence; if a verb takes a subject and two objects, we may have 'X/X+SUB/L_OBJ/R_OBJ/R'. The number of supertags of Model 2 is 312.

Our Model 2 is similar to Model F of Foth et al. (2006) except that they define objects of prepositions and conjunctions as obligatory as well as verbs. However, we define only dependents of verbs because verbs play the most important role for constructing syntactic trees and we would like to decrease the number of supertags.

3 Supertags as Features in a Transition-based Dependency Parser

In this work, we adopt the Easy-First parser of (Goldberg and Elhadad, 2010), a highly-accurate transition-based dependency parser. We describe how we incorporate supertag features in the Easy-First framework, though it can be done similarly

for other transition-based frameworks like left-to-right *arc-eager* and *arc-standard* models (Nivre et al., 2006; Yamada and Matsumoto, 2003).

In the Easy-First algorithm, a dependency tree is constructed by two kinds of actions: ATTACHLEFT(i) and ATTACHRIGHT(i) to a list of partial tree structures p_1,...,p_k initialized with the n words of the sentence w_1,...,w_n. ATTACHLEFT(i) attaches (p_i, p_{+1}) and removes p_{i+1} from the partial tree list. ATTACHRIGHT(i) attaches (p_{i+1}, p_i) and removes p_i from the partial tree list. Features are extracted from the attachment point as well as two neighboring structures: p_{i-2}, p_{i-1}, p_i, p_{i+1}, p_{i+2}, p_{i+3}. Table 2 summarizes the supertag features we extract from this neighborhood; these are appended to the original baseline features based on POS/word in Goldberg and Elhadad (2010).

For a partial tree structure p, features are defined based on information in its head: we use w_p to refer to the surface word form of the head word of p, t_p to refer to the head word's POS tag, and s_p to refer to the head word's supertag. Further, we not only use a supertag as is, but split each supertag into subparts. For instance, the supertag 'ROOT+SUB/L_PRD/R' is split into 'ROOT', 'SUB/L' and 'PRD/R', a supertag representing the supertag head information sh_p, supertag left dependent information sld_p, and supertag right dependent information srd_p.

For the unigram features, we use information within a single partial structure, such as conjunction of head word and its supertag ($w_p s_p$), conjunction of head word's POS tag and its supertag ($t_p s_p$). To consider more context, bigram features look at pairs of partial structures. For each (p, q) pair of structures in p_{i-2}, p_{i-1}, p_i, p_{i+1}, p_{i+2}, we look at e.g. conjunctions of supertags ($s_p s_q$).

Finally, head information of a partial structure and dependent information of another partial structure are combined as "head-dependent features" in order to check for consistency in head-dependent relations. For instance, in Table 1 the supertag for the word 'Black' has head part 'NMOD/R' wanting to attach right and the supertag for the word 'Monday' has dependent part 'L' wanting something to the left; they are likely to be attached by our parser because of the consistency in head-dependent direction. These features are used in conjunction with word and POS-tag.

Model	# tags	Dev	Test
Model1	79	87.81	88.12
Model2	312	87.22	87.13

Table 3: Supertag accuracy evaluated on development and test set. Dev = development set, PTB 22; Test = test set, PTB 23

4 Experiments

To evaluate the effectiveness of supertags as features, we perform experiments on the Penn Treebank (PTB), converted into dependency format with Penn2Malt[1]. Adopting standard approach, we split PTB sections 2-21 for training, section 22 for development and 23 for testing. We assigned POS-tags to the training data by ten-fold jackknifing following Huang and Sagae (2010). Development and test sets are automatically tagged by the tagger trained on the training set.

4.1 Supertagging Experiments

We use the training data set to train a supertagger of each model using Conditional Random Fields (CRF) and the test data set to evaluate the accuracies. We use version 0.12 of CRFsuite[2] for our CRF implementation. First-order transitions, and word/POS of uni, bi and trigrams in a 7-word window surrounding the target word are used as features. Table 3 shows the result of the supertagging accuracies. The supertag accuracies are around 87-88% for both models, suggesting that most of the supertags can be effectively learned by standard CRFs. The tagger takes 0.001 and 0.005 second per sentence for Model 1 and 2 respectively.

In our error analysis, we find it is challenging to assign correct supertags for obligatory dependents of Model 2. In the test set, the number of the supertags encoding obligatory dependents is 5432 and its accuracy is 74.61% (The accuracy of the corresponding supertags in Model 1 is 82.18%). Among them, it is especially difficult to predict the supertags encoding obligatory dependents with a head information of subordination conjunction 'SBAR', such as 'SBAR/L+SUB/L_PRD/R'. The accuracy of such supertags is around 60% (e.g., the accuracy of a supertag 'SBAR/L+SUB/L_PRD/R' is 57.78%), while the supertags encoding dependents with a la-

[1] http://stp.lingfil.uu.se/ nivre/research/Penn2Malt.jar
[2] http://www.chokkan.org/software/crfsuite/

feature	Model1	Model2
baseline	90.25	90.25
+unigram of supertag	90.59	90.76
+bigram of supertag	91.37	91.08
+head-dependent	91.22	91.28

Table 4: Unlabeled attachment scores (UAS) on the development set for each feature template.

Model	UAS	Root	Complete
baseline	90.05	91.10	37.41
Model 1	91.35	93.17	41.35
Model 2	91.23	92.72	41.35

Table 5: Accuracies for English dependency parsing on the test set. UAS = unlabeled attachment score; Root = root attachment score; Complete = the percentage of sentences in which all tokens were assigned their correct heads.

bel 'VC' are assigned almost correctly (e.g., the accuracy of 'VC/L+VC/R' is 97.41%). A verb within a subordinating clause usually has the subordinating conjunction as its head and it tends to be long-range dependency, which is harder to predict. 'VC' represents verb complements. A gerund and a past participle is often a dependent of the immediate front verb, so it is not so difficult to identify the dependency relation.

4.2 Dependency Parsing Experiments

First, we evaluate the effectiveness of the feature templates proposed in Section 3. Following the same procedure as our POS tagger, we first assign supertags to the training data by ten-fold jackknifing, then train our Easy-First dependency parser on these predicted supertags. For development and test sets, we assign supertags based on a supertagger trained on the whole training data.

Table 4 shows the effect of new supertag features on the development data. We start with the baseline features, and incrementally add the unigrams, bigrams, and head-dependent feature templates. For Model 1 we observe that adding unigram features improve the baseline UAS slightly by 0.34% while additionally adding bigram features give larger improvements of 0.78%. On the other hand, for Model 2 unigram features make bigger contribution on improvements by 0.51% than bigram ones 0.32%. One possible explanation is that because each supertag of Model 2 encodes richer syntactic information, an individual tag can make bigger contribution on improvements than Model 1 as a unigram feature. However, since supertags of Model 2 can be erroneous and noisy combination of multiple supertags, such as bigram features, can propagate errors.

Using all features, the accuracy of the accuracy of Model 2 improved further by 0.20%, while Model 1 dropped by 0.15%. It is unclear why Model 1 accuracy dropped, but one hypothesis is that coarse-grained supertags may conflate some head-dependent. The development set UAS for combinations of all features are 91.22% (Model 1) and 91.28% (Model 2), corresponding to 0.97% and 1.03% improvement over the baseline.

Next, we show the parsing accuracies on the test set, using all unigram, bigram, and head-dependents supertag features. The UAS[3], Root attachment scores, and Complete accuracy are shown in Table 5. Both Model 1 and 2 outperform the baseline in all metrics. UAS improvements for both models are statistically significant under the McNemar test, $p < 0.05$ (difference between Model 1 and 2 is not significant). Notably, Model 1 achieves parsing improvements of 1.3% in unlabeled attachment, 2.07% root attachment, and 3.94% in complete accuracy. Comparing Model 1 to baseline, attachment improvements binned by distance to head are as follows: +0.54 F1 for distance 1, +0.81 for distance 2, +2.02 for distance 3 to 6, +2.95 for distance 7 or more, implying supertags are helpful for long distance dependencies.

5 Conclusions

We have demonstrated the effectiveness of supertags as features for English transition-based dependency parsing. In previous work, syntactic information, such as a head and dependents of a word, cannot be used as features before partial tree structures are constructed (Zhang and Nivre, 2011; Goldberg and Elhadad, 2010). By using supertags as features, we can utilize fine-grained syntactic information without waiting for partial trees to be built, and they contribute to improvement of accuracies of English dependency parsing. In future work, we would like to develop parsers that directly integrate supertag ambiguity in the parsing decision, and to investigate automatic pattern mining approaches to supertag design.

[3]For comparison, MaltParser and MSTParser with baseline features is 88.68% and 91.37% UAS respectively

References

Bharat R Ambati, Tejaswini Deoskar and Mark Steedman. 2013. Using CCG categories to improve Hindi dependency parsing. In *Proceedings of ACL*, pages 604-609, Sofia, Bulgaria, August.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Compuational Linguistics*, 25(2):237-265.

Kilian Foth, Tomas By, and Wolfgang Menzel. 2006. Guiding a Constraint Dependency Parser with Supertags. In *Proceedings of COLING/ACL 2006*, pages 289-296, Sydney, Australia, July.

Yoav Goldberg and Michael Elhadad. 2010. An Efficient Algorithm for Easy-First Non-Directional Dependency Parsing. In *Proceedings of HLT/NAACL*, pages 742-750, Los Angeles, California, June.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL*, pages 1077-1086, Uppsala, Sweden, July.

Mitchell. P. Marcus, Beatrice Santorini and Mary Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313-330

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryiğit, Sandra Kübler, Svetoslav, Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95-135

Joakim Nivre, Johan Hall, Jens Nilsson, Gülsen Eryiğit and Svetoslav, Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL*, pages 221-225, New York, USA.

N Okazaki. 2007. CRFsuite: a fast implementation of Conditional Random Fields (CRFs). http://www.chokkan.org/software/crfsuite/.

H Yamada and Y Matsumoto. 2003. Statistical dependency analysis using support vector machines. In *Proceedings of IWPT*, Nancy, France.

Yue Zhang and Joakim Nivre. 2011. Transition-based Dependency Parsing with Rich Non-local Feaures. In *Proceedings of ACL*, pages 188-193, Porland, Oregon, June.

Yao-zhong Zhang, Takuya Matsuzaki and Jun'ichi Tsujii. 2010. A Simple Approach for HPSG Supertagging Using Dependency Information. In *Proceedings of HLT/NAACL*, pages 645-648, Los Angeles, California, June.