HPSG Supertagging: A Sequence Labeling View

Yao-zhong Zhang † **Takuya Matsuzaki** † **Jun'ichi Tsujii**†‡§

† Department of Computer Science, University of Tokyo
‡ School of Computer Science, University of Manchester
§National Centre for Text Mining, UK
{yaozhong.zhang, matuzaki, tsujii}@is.s.u-tokyo.ac.jp

Abstract

Supertagging is a widely used speed-up technique for deep parsing. In another aspect, supertagging has been exploited in other NLP tasks than parsing for utilizing the rich syntactic information given by the supertags. However, the performance of supertagger is still a bottleneck for such applications. In this paper, we investigated the relationship between supertagging and parsing, not just to speed up the deep parser; We started from a sequence labeling view of HPSG supertagging, examining how well a supertagger can do when separated from parsing. Comparison of two types of supertagging model, point-wise model and sequential model, showed that the former model works competitively well despite its simplicity, which indicates the true dependency among supertag assignments is far more complex than the crude first-order approximation made in the sequential model. We then analyzed the limitation of separated supertagging by using a CFG-filter. The results showed that big gains could be acquired by resorting to a light-weight parser.

1 Introduction

Supertagging is an important part of lexicalized grammar parsing. A high performance supertagger greatly reduces the load of a parser and accelerates its speed. A supertag represents a linguistic word category, which encodes syntactic behavior of the word. The concept of supertagging was first proposed for lexicalized tree adjoining grammar (LTAG) (Bangalore and Joshi, 1999) and then extended to other lexicalized grammars, such as combinatory categorial grammar (CCG) (Clark, 2002) and Head-driven phrase structure grammar (HPSG) (Ninomiya et al., 2006). Recently, syntactic information in supertags has been exploited for NLP tasks besides parsing, such as NP chunking (Shen and Joshi, 2003), semantic role labeling (Chen and Rambow, 2003) and machine translation (Hassan et al., 2007). Supertagging serves there as an implicit and convenient way to incorporate rich syntactic information in those tasks.

Improving the performance of supertagging can thus benefit these two aspects: as a preprocessor for deep parsing and as an independent, alternative technique for "almost" parsing. However, supertags are derived from a grammar and thus have a strong connection to parsing. To further improve the supertagging accuracy, the relation between supertagging and parsing is crucial. With this motivation, we investigate how well a sequence labeling model can do when it is separated from a parser, and to what extent the ignorance of long distance dependencies in the sequence labeling formulation affects the supertagging results.

Specifically, we evaluated two different types of supertagging model, point-wise model and sequential model, for HPSG supertagging. CFG-filter was then used to empirically evaluate the effect of long distance dependencies in supertagging. The point-wise model achieved competitive result of 92.53% accuracy on WSJ-HPSG treebank with fast training speed, while the sequential model augmented with supertag edge features did not give much further improvement over the point-wise model. Big gains acquired by using CFG-filter indicates that further improvement may be achieved by resorting to a light-weight parser.

2 HPSG Supertags

HPSG (Pollard and Sag, 1994) is a kind of lexicalized grammar. In HPSG, many lexical entries are used to express word-specific characteristics, while only small amount of rule schemas are used to describe general constructions. A supertag in HPSG corresponds to a template of lexical entry. For example, one possible supertag for "big" is "[<ADJP>]N_lxm", which indicates that the syntactic category of "big" is adjective and it modifies a noun to its right. The number of supertags is generally much larger than the number of labels used in other sequence labeling tasks; Comparing to 45 POS tags used in PennTreebank, the HPSG grammar used in our experiments includes 2,308 supertags. Because of this, it is often very hard or even impossible to apply computationary demanding methods to HPSG supertagging.

3 Perceptron and Bayes Point Machine

Perceptron is an efficient online discriminative training method. We used perceptron with weight-averaging (Collins, 2002) as the basis of our supertagging model. We also use perceptron-based Bayes point machine (BPM) (Herbrich et al., 2001) in some of the experiments. In short, a BPM is an average of a number of averaged perceptrons' weights. We use average of 10 averaged perceptrons, each of which is trained on a different random permutation of the training data.

3.1 Formulation

Here we follow the definition of Collins' perceptron to learn a mapping from the input space $(w, p) \in W \times P$ to the supertag space $s \in S$. We use function **GEN(w,p)** to indicate all candidates given input (w, p). Feature function **f** maps a training sample $(w, p, s) \in W \times P \times S$ to a point in the feature space R^d. To get feature weights $\alpha \in R^d$ of feature function, we used the averaged perceptron training method described in (Collins, 2002), and the average of its 10 different runs (i.e., BPM). For decoding, given an input (w, p) and a vector of feature weights α, we want to find an output s which satisfies:

$$F(w,p) = \underset{s \in \textbf{GEN(w, p)}}{argmax} \quad \alpha \cdot f(w,p,s)$$

For the input (w, p), we treat it in two fashions: one is (w, p) representing a single word and a POS tag. Another is (w, p) representing whole word and POS tags sequence. We call them point-wise model and sequential model respectively. Viterbi algorithm is used for decoding in sequential model.

template type	template
Word	w_i,w_{i-1},w_{i+1}, $w_{i-1}\&w_i$, $w_i\&w_{i+1}$
POS	p_i, p_{i-1}, p_{i-2}, p_{i+1}, p_{i+2}, $p_{i-1}\&p_i$, $p_{i-2}\&p_{i-1}$, $p_{i-1}\&p_{i+1}$, $p_i\&p_{i+1}$, $p_{i+1}\&p_{i+2}$
Word-POS	$p_{i-1}\&w_i$, $p_i\&w_i$, $p_{i+1}\&w_i$
Supertag†	s_{i-1} , $s_{i-2}\&s_{i-1}$
Substructure	$\{ss_{i,1}, ..., ss_{i,N}\}\times$ Word $\{ss_{i,1}, ..., ss_{i,N}\}\times$ POS $\{ss_{i,1}, ..., ss_{i,N}\}\times$ Word-POS $\{ss_{i-1,1}, ..., ss_{i-1,N}\}\times \{ss_{i,1}, ..., ss_{i,N}\}^\dagger$

Table 1: Feature templates for point-wise model and sequential model. Templates with † are only used by sequential model. $ss_{i,j}$ represents j-th substructure of supertag at i. For briefness, s_i is omitted for each template. "$\times$" means set-product. e.g., {a,b}$\times${A,B}={a&A,a&B,b&A,b&B}

3.2 Features

Feature templates are listed in Table 1. To make the results comparable with previous work, we adopt the same feature templates as Matsuzaki et. al. (2007). For sequential model, supertag contexts are added to the features. Because of the large number of supertags, those supertag edge features could be very sparse. To alleviate this sparseness, we extracted sub-structures from the lexical template of each supertag, and use them for making generalized node/edge features as shown in Table 1. The sub-structures we used include subcategorization frames (e.g., subject=NP, object=NP_PP), direction and category of modifiee phrase (e.g., mod_left=VP), voice and tense of a verb (e.g., passive_past).

3.3 CFG-filter

Long distance dependencies are also encoded in supertags. For example, when a transitive verb gets assigned a supertag that specifies it has a PP-object, in most cases a preposition to its right must be assigned an argument (not adjunct) supertag, and vice versa. Such kind of long distance context information might be important for supertag disambiguation, but is not easy to incorporate into a sequence labeling model separated from a parser.

To examine the limitation of supertagging separated from a parser, we used CFG-filter as an ap-

Model Name	Acc%
PW-AP	92.29
SEQ-AP	92.53
PW-AP+CFG	**93.57**
SEQ-AP+CFG	**93.68**

Table 2: Averaged 10-cross validation of averaged perceptron on Section 02-21.

proximation of an HPSG parser. We firstly created a CFG that approximates the original HPSG grammar, using the iterative method by Kiefer and Krieger (2000). Given the supertags as preterminals, the approximating CFG was then used for finding a maximally scored sequence of supertags which satisfies most of the grammatical constraints in the original HPSG grammar (Matsuzaki et al., 2007). By comparing the supertagging results before and after CFG-filtering, we can quantify how many errors are caused by ignorance of the long-range dependencies in supertagger.

4 Experiments and Analysis

We conducted experiments on WSJ-HPSG treebank corpus (Miyao, 2006), which was semi-automatically converted from the WSJ portion of PennTreebank. The number of training iterations was set to 5 for all models. Gold-standard POS tags are used as input. The performance is evaluated by accuracy[1] and speed of supertagging on an AMD Opteron 2.4GHz server.

Table 2 shows the averaged results of 10-fold cross-validation of averaged perceptron (AP) models[2] on section 02-21. We can see the difference between point-wise AP model and sequential AP model is small (0.24%). It becomes even smaller after CFG-filtering (0.11%). Table 3 shows the supertagging accuracy on section 22 based on BPM. Although not statistically significantly different from previous ME model (Matsuzaki et al., 2007), point-wise model (PW-BPM) achieved competitive result 92.53% with faster training. In addition, 0.27% and 0.29% gains were brought by using BPM from PW-AP (92.26%) and PW-SEQ (92.54%) with P-values less than 0.05.

The improvement by using sequential models (PW-AP→SEQ-AP: 0.24%, PW-BPM→SEQ-BPM: 0.3%, statistically significantly different),

[1]"UNK" supertags are ignored in evaluation as previous.

[2]For time limitation, cross validation for BPM was not conducted.

Model Name	Acc%	Training/ Testing Time ‡
ME (Matsuzaki 07')	92.45	≈ 3h / 12s
PW-BPM	92.53	**285s / 10s**
SEQ-BPM	92.83	1721s / 13s
PW-BPM+SUB	92.68	1275s / 25s
SEQ-BPM+SUB	92.99	9468s / 107s
PW-BPM+CFG	**93.60**	**285s / 78s**
SEQ-BPM+CFG	**93.70**	1721s / 195s
PW-BPM+SUB+CFG	**93.72**	1275s / 170s
SEQ-BPM+SUB+CFG	**93.88**	9468s / 1011s

Table 3: Supertagging accuracy and training& testing speed on section 22. (‡) Test time was calculated on totally 1648 sentences.

compared to point-wise models, were not so large, but the training time was around 6 times longer. We think the reason is twofold. First, as previous research showed, POS sequence is very informative in supertagging (Clark, 2004). A large amount of local syntactic information can be captured in POS tags of surrounding words, although a few long-range dependencies are of course not. Second, the number of supertags is large and the supertag edge features used in sequential model are inevitably suffered from data sparseness. To alleviate this, we extracted sub-structure from lexical templates (i.e., lexical items corresponding to supertags) to augment the supertag edge features, but only got 0.16% improvement (SEQ-BPM+SUB). Furthermore, we also got 0.15% gains with P-value less than 0.05 by incorporating the sub-structure features into point-wise model (PW-BPM+SUB). We hence conclude that the contribution of the first-order edge features is not large in sequence modeling for HPSG supertagging.

As we explained in Section 3.3, sequence labeling models have inherent limitation in the ability to capture long distance dependencies between supertags. This kind of ambiguity could be easier to solve in a parser. To examine this, we added CFG-filter which works as an approximation of a full HPSG parser, after the sequence labeling model. As expected, there came big gains of 1.26% (from PW-AP to PW-AP+CFG) and 1.15% (from PW-BPM to PW-BPM+CFG). Even for the sequential model we also got 1.15% (from SEQ-AP to SEQ-AP+CFG) and 0.87% (from SEQ-BPM to SEQ-BPM+CFG) respectively. All these models were statistically significantly different from original ones.

We also gave error analysis on test results. Comparing SEQ-AP with SEQ-AP+CFG, one of the most frequent types of "correct supertag" by the CFG-filter was for word "and", wherein a supertag for NP-coordination ("NP and NP") was corrected to one for VP-coordination ("VP and VP" or "S and S"). It means the disambiguation between the two coordination type is difficult for supertaggers, presumably because they looks very similar with a limited length of context since the sequence of the NP-object of left conjunct, "and", the NP subject of right conjunct looks very similar to a NP coordination. The different assignments by SEQ-AP+CFG from SEQ-AP include 725 right corrections, while it changes 298 correct predictions by SEQ-AP to wrong assignments. One possible reason for some of "wrong correction" is related to the approximation of grammar. But this gives clue that for supertagging task: just using sequence labeling models is limited, and we can resort to use some light-weight parser to handle long distance dependencies.

Although some of the ambiguous supertags could be left for deep parsing, like multi-tagging technique (Clark, 2004), we also consider the tasks where supertags can be used while conducting deep parsing is too computationally costly. Alternatively, focusing on supertagging, we could treat it as a sequence labeling task, while a consequent light-weight parser is a disambiguator with long distance constraint.

5 Conclusions

In this paper, through treating HPSG supertagging in a sequence labeling way, we examined the relationship between supertagging and parsing from an angle. In experiment, even for sequential models, CFG-filter gave much larger improvement than one gained by switching from a point-wise model to a sequential model. The accuracy improvement given by the CFG-filter suggests that we could gain further improvement by combining a supertagger with a light-weight parser.

Acknowledgments

Thanks to the anonymous reviewers for valuable comments. The first author was partially supported by University of Tokyo Fellowship (UT-Fellowship). This work was partially supported by Grant-in-Aid for Specially Promoted Research and Special Coordination Funds for Promoting Science and Technology (MEXT, Japan).

References

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25:237–265.

John Chen and Owen Rambow. 2003. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *Proceedings of EMNLP-2003*, pages 41–48.

Stephen Clark. 2002. Supertagging for combinatory categorial grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+ 6)*, pages 19–24.

Stephen Clark. 2004. The importance of supertagging for wide-coverage ccg parsing. In *Proceedings of COLING-04*, pages 282–288.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. pages 1–8.

Hany Hassan, Mary Hearne, and Andy Way. 2007. Supertagged phrase-based statistical machine translation. In *Proceedings of ACL 2007*, pages 288–295.

Ralf Herbrich, Thore Graepel, and Colin Campbell. 2001. Bayes point machines. *Journal of Machine Learning Research*, 1:245–279.

Bernd Kiefer and Hans-Ulrich Krieger. 2000. A context-free approximation of head-driven phrase structure grammar. In *Proceedings of IWPT-2000*, pages 135–146.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Efficient hpsg parsing with supertagging and cfg-filtering. In *Proceedings of IJCAI-07*, pages 1671–1676.

Yusuke Miyao. 2006. *From Linguistic Theory to Syntactic Analysis: Corpus-Oriented Grammar Development and Feature Forest Model*. Ph.D. Dissertation, The University of Tokyo.

Takashi Ninomiya, Yoshimasa Tsuruoka, Takuya Matsuzaki, and Yusuke Miyao. 2006. Extremely lexicalized models for accurate and fast hpsg parsing. In *Proceedings of EMNLP-2006*, pages 155–163.

Carl Pollard and Ivan A. Sag. 1994. *Head-driven Phrase Structure Grammar*. University of Chicago / CSLI.

Libin Shen and Aravind K. Joshi. 2003. A snow based supertagger with application to np chunking. In *Proceedings of ACL 2003*, pages 505–512.