A SNoW based Supertagger with Application to NP Chunking

Libin Shen and **Aravind K. Joshi**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
{libin,joshi}@linc.cis.upenn.edu

Abstract

Supertagging is the tagging process of assigning the correct elementary tree of LTAG, or the correct supertag, to each word of an input sentence[1]. In this paper we propose to use supertags to expose syntactic dependencies which are unavailable with POS tags. We first propose a novel method of applying Sparse Network of Winnow (SNoW) to sequential models. Then we use it to construct a supertagger that uses long distance syntactical dependencies, and the supertagger achieves an accuracy of 92.41%. We apply the supertagger to NP chunking. The use of supertags in NP chunking gives rise to almost 1% absolute increase (from 92.03% to 92.95%) in F-score under Transformation Based Learning(TBL) frame. The surpertagger described here provides an effective and efficient way to exploit syntactic information.

1 Introduction

In Lexicalized Tree-Adjoining Grammar (LTAG) (Joshi and Schabes, 1997; XTAG-Group, 2001), each word in a sentence is associated with an elementary tree, or a supertag (Joshi and Srinivas, 1994). Supertagging is the process of assigning the correct supertag to each word of an input sentence. The following two facts make supertagging attractive. Firstly supertags encode much more syntactical information than POS tags, which makes supertagging a useful pre-parsing tool, so-called, almost parsing (Srinivas and Joshi, 1999). On the other hand, as the term 'supertagging' suggests, the time complexity of supertagging is similar to that of POS tagging, which is linear in the length of the input sentence.

In this paper, we will focus on the NP chunking task, and use it as an application of supertagging. (Abney, 1991) proposed a two-phase parsing model which includes chunking and attaching. (Ramshaw and Marcus, 1995) approached chucking by using Transformation Based Learning(TBL). Many machine learning techniques have been successfully applied to chunking tasks, such as Regularized Winnow (Zhang et al., 2001), SVMs (Kudo and Matsumoto, 2001), CRFs (Sha and Pereira, 2003), Maximum Entropy Model (Collins, 2002), Memory Based Learning (Sang, 2002) and SNoW (Muñoz et al., 1999).

The previous best result on chunking in literature was achieved by Regularized Winnow (Zhang et al., 2001), which took some of the parsing results given by an English Slot Grammar-based parser as input to the chunker. The use of parsing results contributed 0.62% absolute increase in F-score. However, this approach conflicts with the purpose of chunking. Ideally, a chunker geneates n-best results, and an attacher uses chunking results to construct a parse.

The dilemma is that syntactic constraints are useful in the chunking phase, but they are unavailable until the attaching phase. The reason is that POS tags are not a good labeling system to encode enough linguistic knowledge for chunking. However another labeling system, supertagging, can provide a great deal of syntactic information.

In an LTAG, each word is associated with a set of possible elementary trees. An LTAG parser assigns the correct elementary tree to each word of a sentence, and uses the elementary trees of all the words to build a parse tree for the sentence. Elementary

[1]By the correct supertag we mean the supertag that an LTAG parser would assign to a word in a sentence.

trees, which we call supertags, contain more information than POS tags, and they help to improve the chunking accuracy.

Although supertags are able to encode long distance dependence, supertaggers trained with local information in fact do not take full advantage of complex information available in supertags.

In order to exploit syntactic dependencies in a larger context, we propose a new model of supertagging based on Sparse Network of Winnow (SNoW) (Roth, 1998). We also propose a novel method of applying SNoW to sequential models in a way analogous to the Projection-base Markov Model (PMM) used in (Punyakanok and Roth, 2000). In contrast to PMM, we construct a SNoW classifier for each POS tag. For each word of an input sentence, its POS tag, instead of the supertag of the previous word, is used to select the corresponding SNoW classifier. This method helps to avoid the sparse data problem and forces SNoW to focus on difficult cases in the context of supertagging task. Since PMM suffers from the *label bias problem* (Lafferty et al., 2001), we have used two methods to cope with this problem. One method is to skip the local normalization step, and the other is to combine the results of left-to-right scan and right-to-left scan.

We test our supertagger on both the hand-coded supertags used in (Chen et al., 1999) as well as the supertags extracted from Penn Treebank(PTB) (Marcus et al., 1994; Xia, 2001). On the dataset used in (Chen et al., 1999), our supertagger achieves an accuracy of 92.41%.

We then apply our supertagger to NP chunking. The purpose of this paper is to find a better way to exploit syntactic information which is useful in NP chunking, but not the machine learning part. So we just use TBL, a well-known algorithm in the community of text chunking, as the machine learning tool in our research. Using TBL also allows us to easily evaluate the contribution of supertags with respect to Ramshaw and Marcus's original work, the de facto baseline of NP chunking. The use of supertags with TBL can be easily extended to other machine learning algorithms.

We repeat Ramshaw and Marcus' Transformation Based NP chunking (Ramshaw and Marcus, 1995) algorithm by substituting supertags for POS tags in the dataset. The use of supertags gives rise to almost 1% absolute increase (from 92.03% to 92.95%) in F-score under Transformation Based Learning(TBL) frame. This confirms our claim that using supertagging as a labeling system helps to increase the overall performance of NP Chunking. The supertagger presented in this paper provides an opportunity for advanced machine learning techniques to improve their performance on chunking tasks by exploiting more syntactic information encoded in the supertags.

2 Supertagging and NP Chunking

In (Srinivas, 1997) trigram models were used for supertagging, in which Good-Turing discounting technique and Katz's back-off model were employed. The supertag for a word was determined by the lexical preference of the word, as well as by the contextual preference of the previous two supertags. The model was tested on WSJ section 20 of PTB, and trained on section 0 through 24 except section 20. The accuracy on the test data is 91.37%[2].

In (Srinivas, 1997), supertagging was used for NP chunking and it achieved an F-score of 92.4%. (Chen, 2001) reported a similar result with a trigram supertagger. In their approaches, they first supertagged the test data and then uesd heuristic rules to detect NP chunks. But it is hard to say whether it is the use of supertags or the heuristic rules that makes their system achieve the good results.

As a first attempt, we use *fast TBL* (Ngai and Florian, 2001), a TBL program, to repeat Ramshaw and Marcus' experiment on the standard dataset. Then we use Srinivas' supertagger (Srinivas, 1997) to supertag both the training and test data. We run the *fast TBL* for the second round by using supertags instead of POS tags in the dataset. With POS tags we achieve an F-score of 92.01%, but with supertags we only achieve an F-score of 91.66%. This is not surprising becuase Srinivas' supertag was only trained with a trigram model. Although supertags are able to encode long distance dependence, supertaggers trained with local information in fact do not take full advantage of their strong capability. So we must use long distance dependencies to train supertaggers to take full advantage of the information in supertags.

[2]This number is based on footnote 1 of (Chen et al., 1999). A few supertags were grouped into equivalence classes for evaluation

The trigram model often fails in capturing the co-occurrence dependence between a head word and its dependents. Consider the phrase ”will *join* the board *as* a nonexecutive director”. The occurrence of *join* has influence on the lexical selection of *as*. But *join* is outside the window of trigram. (Srinivas, 1997) proposed a head trigram model in which the lexical selection of a word depended on the supertags of the previous two *head* words , instead of the supertags of the two words immediately leading the word of interest. But the performance of this model was worse than the traditional trigram model because it discarded local information.

(Chen et al., 1999) combined the traditional trigram model and head trigram model in their trigram *mixed* model. In their model, context for the current word was determined by the supertag of the previous word and context for the previous word according to 6 manually defined rules. The *mixed* model achieved an accuracy of 91.79% on the same dataset as that of (Srinivas, 1997). In (Chen et al., 1999), three other models were proposed, but the *mixed* model achieved the highest accuracy. In addition, they combined all their models with *pairwise voting*, yielding an accuracy of 92.19%.

The *mixed* trigram model achieves better results on supertagging because it can capture both local and long distance dependencies to some extent. However, we think that a better way to find useful context is to use machine learning techniques but not define the rules manually. One approach is to switch to models like PMM, which can not only take advantage of generative models with the Viterbi algorithm, but also utilize the information in a larger contexts through flexible feature sets. This is the basic idea guiding the design of our supertagger.

3 SNoW

Sparse Network of Winnow (SNoW) (Roth, 1998) is a learning architecture that is specially tailored for learning in the presence of a very large number of features where the decision for a single sample depends on only a small number of features. Furthermore, SNoW can also be used as a general purpose multi-class classifier.

It is noted in (Muñoz et al., 1999) that one of the important properites of the sparse architecture of SNoW is that the complexity of processing an example depends only on the number of features active in it, n_a, and is independent of the total number of features, n_t, observed over the life time of the system and this is important in domains in which the total number of features in very large, but only a small number of them is active in each example.

As far as supertagging is concerned, word context forms a very large space. However, for each word in a given sentence, only a small part of features in the space are related to the decision on supertag. Specifically the supertag of a word is determined by the appearances of certain words, POS tags, or supertags in its context. Therefore SNoW is suitable for the supertagging task.

Supertagging can be viewed in term of the sequential model, which means that the selection of the supertag for a word is influenced by the decisions made on the previous few words. (Punyakanok and Roth, 2000) proposed three methods of using classifiers in sequential inference, which are HMM, PMM and CSCL. Among these three models, PMM is the most suitable for our task. The basic idea of PMM is as follows.

Given an observation sequence O, we find the most likely state sequence S given O by maximizing

$$\begin{aligned} P(S|O) &= [\prod_{t=2}^{n} P(s_t|s_1,...,s_{t-1},O)]P_1(s_1|o_1) \\ &= [\prod_{t=2}^{n} P(s_t|s_{t-1},o_t)]P_1(s_1|o_1) \qquad (1) \end{aligned}$$

In this model, the output of SNoW is used to estimate $P(s|s',o)$ and $P_1(s|o)$, where s is the current state, s' is the previous state, and o is the current observation. $P(s|s',o)$ is separated to many sub-functions $P_{s'}(s|o)$ according to previous state s'. In practice, $P_{s'}(s|o)$ is estimated in a wider window of the observed sequence, instead of o only. Then the problem is how to map the SNoW results into probabilities. In (Punyakanok and Roth, 2000), the sigmoid $1/(1+e^{-(act-T)})$ is defined as *confidence*, where T is the threshold for SNoW, *act* is the dot product of the weight vector and the example vector. The *confidence* is normalized by summing to 1 and used as the distribution mass $P_{s'}(s|o)$.

4 Modeling Supertagging

4.1 A Novel Sequential Model with SNoW

Firstly we have to decide how to treat POS tags. One approach is to assign POS tags at the same time that we do supertagging. The other approach is to assign POS tags with a traditional POS tagger first, and then use them as input to the supertagger. Supertagging an unknown word becomes a problem for supertagging due to the huge size of the supertag set, Hence we use the second approach in our paper. We first run the Brill POS tagger (Brill, 1995) on both the training and the test data, and use POS tags as part of the input.

Let $W = w_1 w_2 ... w_n$ be the sentence, $Q = q_1 q_2 ... q_n$ be the POS tags, and $T = t_1 t_2 ... t_n$ be the supertags respectively. Given W, Q, we can find the most likely supertag sequence T given W, Q by maximizing

$$P(T|W,Q) = [\prod_{i=2}^{n} P(t_i|t_{1...i-1}, W, Q)] P_1(t_1|w_1, q_1)$$

Analogous to PMM, we decompose $P(t_i|t_{1...i-1}, W, Q)$ into sub-classifiers. However, in our model, we divide it with respect to POS tags as follows

$$P(t_i|t_{1...i-1}, W, Q) \equiv P_{q_i}(t_i|t_{1...i-1}, W, Q) \qquad (2)$$

There are several reasons for decomposing $P(t_i|t_{1...i-1}, W, Q)$ with respect to the POS tag of the current word, instead of the supertag of the previous word.

- To avoid sparse-data problem. There are 479 supertags in the set of hand-coded supertags, and almost 3000 supertags in the set of supertags extracted from Penn Treebank.
- Supertags related to the same POS tag are more difficult to distinguish than supertags related to different POS tags. Thus by defining a classifier on the POS tag of the current word but not the POS tag of the previous word forces the learning algorithm to focus on difficult cases.
- Decomposition of the probability estimation can decrease the complexity of the learning algorithm and allows the use of different parameters for different POS tags.

For each POS q, we construct a SNoW classifier M_q to estimate distribution $P_q(t|t', W, Q)$ according to the previous supertags t'. Following the estimation of distribution function in (Punyakanok and Roth, 2000), we define confidence with a sigmoid

$$C_q(t|t', W, Q) \equiv \frac{1}{1 + \alpha\, e^{-(M_q(t|t', W, Q) - s)}}, \qquad (3)$$

where s is the threshold of M_q, and α is set to 1.

The distribution mass is then defined with normalized confidence

$$P_q(t|t', W, Q) \equiv \frac{C_q(t|t', W, Q)}{\sum_t C_q(t|t', W, Q)} \qquad (4)$$

4.2 Label Bias Problem

In (Lafferty et al., 2001), it is shown that PMM and other non-generative finite-state models based on next-state classifiers share a weakness which they called the *label bias problem*: the transitions leaving a given state compete only against each other, rather than against all other transitions in the model. They proposed Conditional Random Fields (CRFs) as solution to this problem.

(Collins, 2002) proposed a new algorithm for parameter estimation as an alternate to CRF. The new algorithm was similar to maximum-entropy model except that it skipped the local normalization step. Intuitively, it is the local normalization that makes distribution mass of the transitions leaving a given state incomparable with all other transitions.

It is noted in (Muñoz et al., 1999) that SNoW's output provides, in addition to the prediction, a robust confidence level in the prediction, which enables its use in an inference algorithm that combines predictors to produce a coherent inference. In that paper, SNoW's output is used to estimate the probability of *open* and *close* tags. In general, the probability of a tag can be estimated as follows

$$P_q(t|t', W, Q) \equiv \frac{M_q(t|t', W, Q) - s}{\sum_t (M_q(t|t', W, Q) - s)}, \qquad (5)$$

as one of the anonymous reviewers has suggested.

However, this makes probabilities comparable only within the transitions of the same history t'. An alternative to this approach is to use the SNoW's output directly in the prediction combination, which makes transitions of different history comparable, since the SNoW's output provides a robust confidence level in the prediction. Furthermore, in order to make sure that the confidences are not too sharp, we use the *confidence* defined in (3).

In addition, we use two supertaggers, one scans from left to right and the other scans from right to left. Then we combine the results via *pairwise voting* as in (van Halteren et al., 1998; Chen et al., 1999) as the final supertag. This approach of voting also helps to cope with the *label bias problem*.

4.3 Contextual Model

$P_q(t|t', W, Q)$ is estimated within a 5-word window plus two head supertags before the current word. For each word w_i, the basic features are $W = w_{i-2,\ldots,i+2}$, $Q = q_{i-2,\ldots,i+2}$, $t' = t_{i-2,i-1}$ and $hd_{-2,-1}$, the two head supertags before the current word. Thus

$$\begin{aligned} & P_{q_i}(t_i|t_{1\ldots i-1}, W, Q) \\ = \; & P_{q_i}(t_i|t_{i-2,i-1}, w_{i-2\ldots i+2}, q_{i-2\ldots i+2}, hd_{-2,-1}) \end{aligned}$$

A basic feature is called *active* for word w_i if and only if the corresponding word/POS-tag/supertag appears at a specified place around w_i. For our SNoW classifiers we use unigram and bigram of basic features as our feature set. A feature defined as a bigram of two basic features is active if and only if the two basic features are both active. The value of a feature of w_i is set to 1 if this feature is active for w_i, or 0 otherwise.

4.4 Related Work

(Chen, 2001) implemented an MEMM model for supertagging which is analogous to the POS tagging model of (Ratnaparkhi, 1996). The feature sets used in the MEMM model were similar to ours. In addition, prefix and suffix features were used to handle rare words. Several MEMM supertaggers were implemented based on distinct feature sets.

In (Muñoz et al., 1999), SNoW was used for text chunking. The IOB tagging model in that paper was similar to our model for supertagging, but there are some differences. They did not decompose the SNoW classifier with respect to POS tags. They used two-level deterministic (beam-width=1) search, in which the second level IOB classifier takes the IOB output of the first classifier as input features.

5 Experimental Evaluation and Analysis

In our experiments, we use the default settings of the SNoW promotion parameter, demotion parameter and the threshold value given by the SNoW system. We train our model on the training data for 2 rounds, only counting the features that appear for at least 5 times. We skip the normalization step in test, and we use beam search with the width of 5.

In our first experiment, we use the same dataset as that of (Chen et al., 1999) for our experiments. We use WSJ section 00 through 24 expect section 20 as training data, and use section 20 as test data. Both training and test data are first tagged by Brill's POS tagger (Brill, 1995). We use the same pairwise voting algorithm as in (Chen et al., 1999). We run supertagging on the training data and use the supertagging result to generate the mapping table used in pairwise voting.

The SNoW supertagger scanning from left to right achieves an accuracy of 92.02%, and the one scanning from right to left achieves an accuracy of 91.43%. By combining the results of these two supertaggers with pairwise voting, we achieve an accuracy of 92.41%, an error reduction of 2.1% compared to 92.25%, the best supertagging result to date (Chen, 2001). Table 1 shows the comparison with previous work.

Our algorithm, which is coded in Java, takes about 10 minutes to supertag the test data with a P3 1.13GHz processor. However, in (Chen, 2001), the accuracy of 92.25% was achieved by a Viterbi search program that took about 5 days to supertag the test data. The counterpart of our algorithm in (Chen, 2001) is the beam search on Model 8 with width of 5, which is the same as the beam width in our algorithm. Compared with this program, our algorithm achieves an error reduction of 7.1%.

(Chen et al., 1999) achieved an accuracy of 92.19% by combination of 5 distinct supertaggers. However, our result is achieved by combining outputs of two homogeneous supertaggers, which only differ in scan direction.

Our next experiment is with the set of supertags abstracted from PTB with Fei Xia's LexTract (Xia, 2001). Xia extracted an LTAG-style grammar from PTB, and repeated Srinivas' experiment (Srinivas, 1997) on her supertag set. There are 2920 elemen-

model	acc%
Srinivas(97) trigram	91.37
Chen(99) trigram mix	91.79
Chen(99) voting	92.19
Chen(01) width=5	91.83
Chen(01) Viterbi	92.25
SNoW left-to-right	92.02
SNoW right-to-left	91.43
SNoW	**92.41**

Table 1: Comparison with previous work. Training data is WSJ section 00 thorough 24 except section 20 of PTB. Test data is WSJ section 20. Size of tag set is 479. acc% = percentage of accuracy. The number of Srinivas(97) is based on footnote 1 of (Chen et al., 1999). The number of Chen(01) width=5 is the result of a beam search on Model 8 with the width of 5.

model	acc%(22)	acc%(23)
Xia(01) trigram	83.60	84.41
SNoW left-to-right	86.01	86.27

Table 2: Results on auto-extracted LTAG grammar. Training data is WSJ section 02 thorough 21 of PTB. Test data is WSJ section 22 and 23. Size of supertag set is 2920. acc% = percentage of accuracy.

tary trees in Xia's grammar G_2, so that the supertags are more specialized and hence there is much more ambiguity in supertagging. We have experimented with our model on G_2 and her dataset. We train our left-to-right model on WSJ section 02 through 21 of PTB, and test on section 22 and 23. We achieve an average error reduction of 13.0%. The reason why the accuracy is rather low is that systems using G_2 have to cope with much more ambiguities due the large size of the supertag set. The results are shown in Table 2.

We test on both normalized and unnormalized models with both hand coded supertag set and auto-extracted supertag set. We use the left-to-right SNoW model in these experiments. The results in Table 3 show that skipping the local normalization improves performance in all the systems. The effect of skipping normalization is more significant on auto-extracted tags. We think this is because sparse data is more vulnerable to the label bias problem.

tag set	size	norm?	acc%(20/22/23)
auto	2920	yes	NA / 85.77 / 85.98
auto	2920	no	NA / 86.01 / 86.27
hand	479	yes	91.98 / NA / NA
hand	479	no	92.02 / NA / NA

Table 3: Experiments on normalized and unnormalized models using left-to-right SNoW supertagger. size = size of the tag set. norm? = normalized or not. acc% = percentage of accuracy on section 20, 22 and 23. auto = auto-extracted tag set. hand = hand coded tag set.

6 Application to NP Chunking

Now we come back to the NP chunking problem. The standard dataset of NP chunking consists of WSJ section 15-18 as train data and section 20 as test data. In our approach, we substitute the supertags for the POS tags in the dataset. The new data look as follows.

For	B_Pnxs	O
the	B_Dnx	I
nine	B_Dnx	I
months	A_NXN	I

The first field is the word, the second is the supertag of the word, and the last is the IOB tag.

We first use the *fast TBL* (Ngai and Florian, 2001), a Transformation Based Learning algorithm, to repeat Ramshaw and Marcus' experiment, and then apply the same program to our new dataset. Since section 15-18 and section 20 are in the standard data set of NP chunking, we need to avoid using these sections as training data for our supertagger. We have trained another supertagger that is trained on 776K words in WSJ section 02-14 and 21-24, and it is tuned with 44K words in WSJ section 19. We use this supertagger to supertag section 15-18 and section 20. We train an NP Chunker on section 15-18 with *fast TBL*, and test it on section 20.

There is a small problem with the supertag set that we have been using, as far as NP chunking is concerned. Two words with different POS tags may be tagged with the same supertag. For example both determiner (DT) and number (CD) can be tagged with B_Dnx. However this will be harmful in the case

model	A	P	R	F
RM95	-	91.80	92.27	92.03
Brill-POS	97.42	91.83	92.20	92.01
Tri-STAG	97.29	91.60	91.72	91.66
SNoW-STAG	97.66	92.76	92.34	92.55
SNoW-STAG2	97.70	92.86	93.05	**92.95**
GOLD-POS	97.91	93.17	93.51	93.34
GOLD-STAG	98.48	94.74	95.63	95.18

Table 4: Results on NP Chunking. Training data is WSJ section 15-18 of PTB. Test data is WSJ section 20. A = Accuracy of IOB tagging. P = NP chunk Precision. R = NP chunk Recall. F = F-score. Brill-POS = *fast TBL* with Brill's POS tags. Tri-STAG = *fast TBL* with supertags given by Srinivas' trigram-based supertagger. SNoW-STAG = *fast TBL* with supertags given by our SNoW supertagger. SNoW-STAG2 = *fast TBL* with augmented supertags given by our SNoW supertagger. GOLD-POS = *fast TBL* with gold standard POS tags. GOLD-STAG = *fast TBL* with gold standard supertags.

of NP Chunking. As a solution, we use augmented supertags that have the POS tag of the lexical item specified. An augmented supertag can also be regarded as concatenation of a supertag and a POS tag.

For	B_Pnxs(IN)	O
the	B_Dnx(DT)	I
nine	B_Dnx(CD)	I
months	A_NXN(NNS)	I

The results are shown in Table 4. The system using augmented supertags achieves an F-score of 92.95%, or an error reduction of 11.8% below the baseline of using Brill POS tags. Although these two systems are both trained with the same TBL algorithm, we implicitly employ more linguistic knowledge as the learning bias when we train the learning machine with supertags. Supertags encode more syntactical information than POS tag do.

For example, in the sentence *Three leading drug companies ...*, the POS tag of $leading$ is VBG, or present participle. Based on the local context of $leading$, *Three* can be the subject of *leading*. However, the supertag of *leading* is B_An, which represents a modifier of a noun. With this extra information, the chunker can easily solve the ambiguity. We find many instances like this in the test data.

It is important to note that the accuracy of supertag itself is much lower than that of POS tag while the use of supertags helps to improve the overall performance. On the other hand, since the accuracy of supertagging is rather lower, there is more room left for improving.

If we use gold standard POS tags in the previous experiment, we can only achieve an F-score of 93.34%. However, if we use gold standard supertags in our previous experiment, the F-score is as high as 95.18%. This tells us how much room there is for further improvements. Improvements in supertagging may give rise to further improvements in chunking.

7 Conclusions

We have proposed the use of supertags in the NP chunking task in order to use more syntactical dependencies which are unavailable with POS tags. In order to train a supertagger with a larger context, we have proposed a novel method of applying SNoW to the sequential model and have applied it to supertagging. Our algorithm takes advantage of rich feature sets, avoids the sparse-data problem, and forces the learning algorithm to focus on the difficult cases. Being aware of the fact that our algorithm may suffer from the *label bias problem*, we have used two methods to cope with this problem, and achieved desirable results.

We have tested our algorithms on both the hand-coded tag set used in (Chen et al., 1999) and supertags extracted for Penn Treebank(PTB). On the same dataset as that of (Chen et al., 1999), our new supertagger achieves an accuracy of 92.41%. Compared with the supertaggers with the same decoding complexity (Chen, 2001), our algorithm achieves an error reduction of 7.1%.

We repeat Ramshaw and Marcus' Transformation Based NP chunking (Ramshaw and Marcus, 1995) test by substituting supertags for POS tags in the dataset. The use of supertags in NP chunking gives rise to almost 1% absolute increase (from 92.03% to 92.95%) in F-score under Transformation Based Learning(TBL) frame, or an error reduction of 11.8%.

The accuracy of 92.95% with our individual TBL chunker is close to results of POS-tag-based systems using advanced machine learning algorithms, such as 93.34% by voted MBL chunkers (Sang, 2002), 92.8% by SNoW chunker (Muñoz et al., 1999). The benefit of using a supertagger is obvious. The supertagger provides an opportunity for advanced machine learning techniques to improve their performance on chunking tasks by exploiting more syntactic information encoded in the supertags.

To sum up, the supertagging algorithm presented here provides an effective and efficient way to employ syntactic information.

Acknowledgments

We thank Vasin Punyakanok for help on the use of SNoW in sequential inference, John Chen for help on dataset and evaluation methods and comments on the draft. We also thank Srinivas Bangalore and three anonymous reviews for helpful comments.

References

S. Abney. 1991. Parsing by chunks. In *Principle-Based Parsing*. Kluwer Academic Publishers.

E. Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

J. Chen, B. Srinivas, and K. Vijay-Shanker. 1999. New models for improving supertag disambiguation. In *Proceedings of the 9th EACL*.

J. Chen. 2001. *Towards Efficient Statistical Parsing using Lexicalized Grammatical Information*. Ph.D. thesis, University of Delaware.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP 2002*.

A. Joshi and Y. Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69 – 124. Springer.

A. Joshi and B. Srinivas. 1994. Disambiguation of super parts of speech (or supertags): Almost parsing. In *COLING'94*.

T. Kudo and Y. Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of NAACL 2001*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for stgmentation and labeling sequence data. In *Proceedings of ICML 2001*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1994. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

M. Muñoz, V. Punyakanok, D. Roth, and D. Zimak. 1999. A learning approach to shallow parsing. In *Proceedings of EMNLP-WVLC'99*.

G. Ngai and R. Florian. 2001. Transformation-based learning in the fast lane. In *Proceedings of NAACL-2001*, pages 40–47.

V. Punyakanok and D. Roth. 2000. The use of classifiers in sequential inference. In *NIPS'00*.

L. Ramshaw and M. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd WVLC*.

A. Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of EMNLP 96*.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *AAAI'98*.

Erik F. Tjong Kim Sang. 2002. Memory-based shallow parsing. *Journal of Machine Learning Research*, 2:559–594.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of NAACL 2003*.

B. Srinivas and A. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2).

B. Srinivas. 1997. Performance evaluation of supertagging for partial parsing. In *IWPT 1997*.

H. van Halteren, J. Zavrel, and W. Daelmans. 1998. Improving data driven wordclass tagging by system combination. In *Proceedings of COLING-ACL 98*.

F. Xia. 2001. *Automatic Grammar Generation From Two Different Perspectives*. Ph.D. thesis, University of Pennsylvania.

XTAG-Group. 2001. A lexicalized tree adjoining grammar for english. Technical Report 01-03, IRCS, Univ. of Pennsylvania.

T. Zhang, F. Damerau, and D. Johnson. 2001. Text chunking using regularized winnow. In *Proceedings of ACL 2001*.