A Fast, Accurate Deterministic Parser for Chinese

Mengqiu Wang   Kenji Sagae   Teruko Mitamura
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
{mengqiu,sagae,teruko}@cs.cmu.edu

Abstract

We present a novel classifier-based deterministic parser for Chinese constituency parsing. Our parser computes parse trees from bottom up in one pass, and uses classifiers to make shift-reduce decisions. Trained and evaluated on the standard training and test sets, our best model (using stacked classifiers) runs in linear time and has labeled precision and recall above 88% using gold-standard part-of-speech tags, surpassing the best published results. Our SVM parser is 2-13 times faster than state-of-the-art parsers, while producing more accurate results. Our Maxent and DTree parsers run at speeds 40-270 times faster than state-of-the-art parsers, but with 5-6% losses in accuracy.

1 Introduction and Background

Syntactic parsing is one of the most fundamental tasks in Natural Language Processing (NLP). In recent years, Chinese syntactic parsing has also received a lot of attention in the NLP community, especially since the release of large collections of annotated data such as the Penn Chinese Treebank (Xue et al., 2005). Corpus-based parsing techniques that are successful for English have been applied extensively to Chinese. Traditional statistical approaches build models which assign probabilities to every possible parse tree for a sentence. Techniques such as dynamic programming, beam-search, and best-first-search are then employed to find the parse tree with the highest probability. The massively ambiguous nature of wide-coverage statistical parsing,coupled with cubic-time (or worse) algorithms makes this approach too slow for many practical applications.

Deterministic parsing has emerged as an attractive alternative to probabilistic parsing, offering accuracy just below the state-of-the-art in syntactic analysis of English, but running in linear time (Sagae and Lavie, 2005; Yamada and Matsumoto, 2003; Nivre and Scholz, 2004). Encouraging results have also been shown recently by Cheng et al. (2004; 2005) in applying deterministic models to Chinese dependency parsing.

We present a novel classifier-based deterministic parser for Chinese constituency parsing. In our approach, which is based on the shift-reduce parser for English reported in (Sagae and Lavie, 2005), the parsing task is transformed into a succession of classification tasks. The parser makes one pass through the input sentence. At each parse state, it consults a classifier to make shift/reduce decisions. The parser then commits to a decision and enters the next parse state. Shift/reduce decisions are made deterministically based on the local context of each parse state, and no backtracking is involved. This process can be viewed as a greedy search where only one path in the whole search space is considered. Our parser produces both dependency and constituent structures, but in this paper we will focus on constituent parsing.

By separating the classification task from the parsing process, we can take advantage of many machine learning techniques such as classifier ensemble. We conducted experiments with four different classifiers: support vector machines (SVM), Maximum-Entropy (Maxent), Decision Tree (DTree) and memory-based learning (MBL). We also compared the performance of three different classifier ensemble approaches (simple voting, classifier stacking and meta-classifier).

Our best model (using stacked classifiers) runs in linear time and has labeled precision and recall above 88% using gold-standard part-of-speech tags, surpassing the best published results (see Section 5). Our SVM parser is 2-13 times faster than state-of-the-art parsers, while producing more accurate results. Our Maxent and DTree parsers are 40-270 times faster than state-of-the-art parsers, but with 5-6% losses in accuracy.

2 Deterministic parsing model

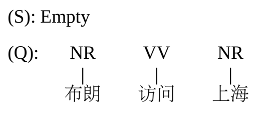
Like other deterministic parsers, our parser assumes input has already been segmented and tagged with part-of-speech (POS) information during a preprocessing step[1]. The main data structures used in the parsing algorithm are a queue and a stack. The input word-POS pairs to be processed are stored in the queue. The stack holds the partial parse trees that are built during parsing. A parse state is represented by the content of the stack and queue.

The classifier makes shift/reduce decisions based on contextual features that represent the parse state. A shift action removes the first item on the queue and puts it onto the stack. A reduce action is in the form of Reduce-{Binary|Unary}-X, where {Binary|Unary} denotes whether one or two items are to be removed from the stack, and X is the label of a new tree node that will be dominating the removed items. Because a reduction is either unary or binary, the resulting parse tree will only have binary and/or unary branching nodes.

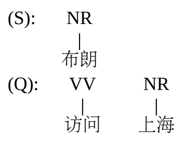
Parse trees are also lexicalized to produce dependency structures. For lexicalization, we used the same head-finding rules reported in (Bikel, 2004). With this additional information, reduce actions are now in the form of Reduce-{Binary |Unary}-X-Direction. The "Direction" tag gives information about whether to take the head-node of the left subtree or the right subtree to be the head of the new tree, in the case of binary reduction. A simple transformation process as described in (Sagae and Lavie, 2005) is employed to convert between arbitrary branching trees and binary trees. This transformation breaks multi-branching nodes down into binary-branching nodes by inserting temporary nodes; temporary nodes are collapsed and removed when we transform a binary tree back into a multi-branching tree.

The parsing process succeeds when all the items in the queue have been processed and there is only one item (the final parse tree) left on the stack. If the classifier returns a shift action when there are no items left on the queue, or a reduce action when there are no items on the stack, the parser fails. In this case, the parser simply combines all the items on the stack into one IP node, and outputs this as a partial parse. Sagae and Lavie (2005) have shown that this algorithm has linear time complexity, assuming that classification takes constant time. The next example illustrates the process for the input "布朗 (Brown) 访问 (visits) 上海 (Shanghai)" that is tagged with the POS sequence "NR (Proper Noun) VV (Verb) NR (Proper Noun)".

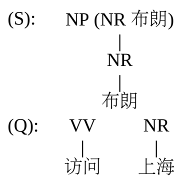
1. In the initial parsing state, the stack (S) is empty, and the queue (Q) holds word and POS tag pairs for the input sentence.



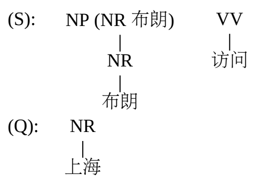
2. The first action item that the classifier gives is a shift action.



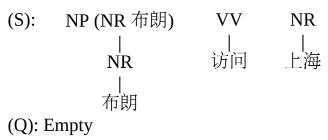
3. The next action is a reduce-Unary-NP, which means reducing the first item on the stack to a NP node. Node (NR 布朗) becomes the head of the new NP node and this information is marked by brackets. The new parse state is:



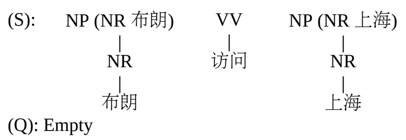
4. The next action is shift.



5. The next action is again shift.



6. The next action is reduce-Unary-NP.

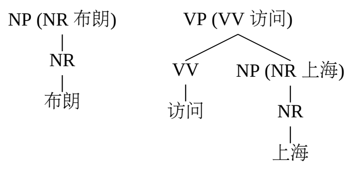


7. The next action is reduce-Binary-VP-Left. The node (VV 访问) will be the head of the

[1]We constructed our own POS tagger based on SVM; see Section 3.3.

new VP node.

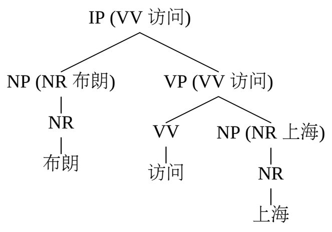
(S):



(Q): Empty

8. The next action is reduce-Binary-IP-Right. Since after the action is performed, there will be only one tree node(IP) left on the stack and no items on the queue, this is the final action. The final state is:

(S):



(Q): Empty

3 Classifiers and Feature Selection

Classification is the key component of our parsing model. We conducted experiments with four different types of classifiers.

3.1 Classifiers

Support Vector Machine: Support Vector Machine is a discriminative classification technique which solves the binary classification problem by finding a hyperplane in a high dimensional space that gives the maximum soft margin, based on the Structural Risk Minimization Principle. We used the TinySVM toolkit (Kudo and Matsumoto, 2000), with a degree 2 polynomial kernel. To train a multi-class classifier, we used the one-against-all scheme.

Maximum-Entropy Classifier: In a Maximum-entropy model, the goal is to estimate a set of parameters that would maximize the entropy over distributions that satisfy certain constraints. These constraints will force the model to best account for the training data (Ratnaparkhi, 1999). Maximum-entropy models have been used for Chinese character-based parsing (Fung et al., 2004; Luo, 2003) and POS tagging (Ng and Low, 2004). In our experiments, we used Le's Maxent toolkit (Zhang, 2004). This implementation uses the Limited-Memory Variable Metric method for parameter estimation. We trained all our models using 300 iterations with no event cut-off, and a Gaussian prior smoothing value of 2. Maxent classifiers output not only a single class label, but also a number of possible class labels and their associated probability estimate.

Decision Tree Classifier: Statistical decision tree is a classic machine learning technique that has been extensively applied to NLP. For example, decision trees were used in the SPATTER system (Magerman, 1994) to assign probability distribution over the space of possible parse trees. In our experiment, we used the C4.5 decision tree classifier, and ignored lexical features whose counts were less than 7.

Memory-Based Learning: Memory-Based Learning approaches the classification problem by storing training examples explicitly in memory, and classifying the current case by finding the most similar stored cases (using k-nearest-neighbors). We used the TiMBL toolkit (Daelemans et al., 2004) in our experiment, with $k = 5$.

3.2 Feature selection

For each parse state, a set of features are extracted and fed to each classifier. Features are distributionally-derived or linguistically-based, and carry the context of a particular parse state. When input to the classifier, each feature is treated as a contextual predicate which maps an outcome and a context to $true, false$ value.

The specific features used with the classifiers are listed in Table 1.

Sun and Jurafsky (2003) studied the distributional property of rhythm in Chinese, and used the rhythmic feature to augment a PCFG model for a practical shallow parsing task. This feature has the value 1, 2 or 3 for monosyllabic, bi-syllabic or multi-syllabic nouns or verbs. For noun and verb phrases, the feature is defined as the number of words in the phrase. Sun and Jurafsky found that in NP and VP constructions there are strong constraints on the word length for verbs and nouns (a kind of rhythm), and on the number of words in a constituent. We employed these same rhythmic features to see whether this property holds for the Penn Chinese Treebank data, and if it helps in the disambiguation of phrase types. Experiments show that this feature does increase classification accuracy of the SVM model by about 1%.

In both Chinese and English, there are punctuation characters that come in pairs (e.g., parentheses). In Chinese, such pairs are more frequent (quotes, single quotes, and book-name marks). During parsing, we note how many opening punc-

1	A Boolean feature indicates if a closing punctuation is expected or not.
2	A Boolean value indicates if the queue is empty or not.
3	A Boolean feature indicates whether there is a comma separating S(1) and S(2) or not.
4	Last action given by the classifier, and number of words in S(1) and S(2).
5	Headword and its POS of S(1), S(2), S(3) and S(4), and word and POS of Q(1), Q(2), Q(3) and Q(4).
6	Nonterminal label of the root of S(1) and S(2), and number of punctuations in S(1) and S(2).
7	Rhythmic features and the linear distance between the head-words of the S(1) and S(2).
8	Number of words found so far to be dependents of the head-words of S(1) and S(2).
9	Nonterminal label, POS and headword of the immediate left and right child of the root of S(1) and S(2).
10	Most recently found word and POS pair that is to the left of the head-word of S(1) and S(2).
11	Most recently found word and POS pair that is to the right of the head-word of S(1) and S(2).

Table 1: Features for classification

tuations we have seen on the stack. If the number is odd, then feature 2 will have value 1, otherwise 0. A boolean feature is used to indicate whether or not an odd number of opening punctuations have been seen and a closing punctuation is expected; in this case the feature gives a strong hint to the parser that all the items in the queue before the closing punctuation, and the items on the stack after the opening punctuation should be under a common constituent node which begins and ends with the two punctuations.

3.3 POS tagging

In our parsing model, POS tagging is treated as a separate problem and it is assumed that the input has already been tagged with POS. To compare with previously published work, we evaluated the parser performance on automatically tagged data. We constructed a simple POS tagger using an SVM classifier. The tagger makes two passes over the input sentence. The first pass extracts features from the two words and POS tags that came before the current word, the two words following the current word, and the current word itself (the length of the word, whether the word contains numbers, special symbols that separates foreign first and last names, common Chinese family names, western alphabets or dates). Then the tag is assigned to the word according to SVM classifier's output. In the second pass, additional features such as the POS tags of the two words following the current word, and the POS tag of the current word (assigned in the first pass) are used. This tagger had a measured precision of 92.5% for sentences $\leq$ 40 words.

4 Experiments

We performed experiments using the Penn Chinese Treebank. Sections 001-270 (3484 sentences, 84,873 words) were used for training, 271-300 (348 sentences, 7980 words) for development, and 271-300 (348 sentences, 7980 words) for testing. The whole dataset contains 99629 words, which is about 1/10 of the size of the English Penn Treebank. Standard corpus preparation steps were done prior to parsing, so that empty nodes were removed, and the resulting A over A unary rewrite nodes are collapsed. Functional labels of the nonterminal nodes are also removed, but we did not relabel the punctuations, unlike in (Jiang, 2004). Bracket scoring was done by the EVALB program[2], and preterminals were not counted as constituents. In all our experiments, we used labeled recall (LR), labeled precision (LP) and F1 score (harmonic mean of LR and LP) as our evaluation metrics.

4.1 Results of different classifiers

Table 2 shows the classification accuracy and parsing accuracy of the four different classifiers on the development set for sentences $\leq$ 40 words, with gold-standard POS tagging. The runtime (Time) of each model and number of failed parses (Fail) are also shown.

	Classification	Parsing Accuracy				
Model	Accuracy	LR	LP	F1	Fail	Time
SVM	**94.3%**	**86.9%**	**87.9%**	**87.4%**	**0**	3m 19s
Maxent	92.6%	84.1%	85.2%	84.6%	5	0m 21s
DTree1	92.0%	78.8%	80.3%	79.5%	42	**0m 12s**
DTree2	N/A	81.6%	83.6%	82.6%	30	0m 18s
MBL	90.6%	74.3%	75.2%	74.7%	2	16m 11s

Table 2: Comparison of different classifier models' parsing accuracies on development set for sentences $\leq$ 40 words, with gold-standard POS

For the DTree learner, we experimented with two different classification strategies. In our first approach, the classification is done in a single stage (DTree1). The learner is trained for a multi-

[2]http://nlp.cs.nyu.edu/evalb/

class classification problem where the class labels include shift and all possible reduce actions. But this approach yielded a lot of parse failures (42 out of 350 sentences failed during parsing, and partial parse tree was returned). These failures were mostly due to false shift actions in cases where the queue is empty. To alleviate this problem, we broke the classification process down to two stages (DTree2). A first stage classifier makes a binary decision on whether the action is shift or reduce. If the output is reduce, a second-stage classifier decides which reduce action to take. Results showed that breaking down the classification task into two stages increased overall accuracy, and the number of failures was reduced to 30.

The SVM model achieved the highest classification accuracy and the best parsing results. It also successfully parsed all sentences. The Maxent model's classification error rate (7.4%) was 30% higher than the error rate of the SVM model (5.7%), and its F1 (84.6%) was 3.2% lower than SVM model's F1 (87.4%). But Maxent model was about 9.5 times faster than the SVM model. The DTree classifier achieved 81.6% LR and 83.6% LP. The MBL model did not perform well; although MBL and SVM differed in accuracy by only about 3 percent, the parsing results showed a difference of more than 10 percent. One possible explanation for the poor performance of the MBL model is that all the features we used were binary features, and memory-based learner is known to work better with multivalue features than binary features in natural language learning tasks (van den Bosch and Zavrel, 2000).

In terms of speed and accuracy trade-off, there is a 5.5% trade-off in F1 (relative to SVM's F1) for a roughly 14 times speed-up between SVM and two-stage DTree. Maxent is more balanced in the sense that its accuracy was slightly lower (3.2%) than SVM, and was just about as fast as the two-stage DTree on the development set. The high speed of the DTree and Maxent models make them very attractive in applications where speed is more critical than accuracy. While the SVM model takes more CPU time, we show in Section 5 that when compared to existing parsers, SVM achieves about the same or higher accuracy but is at least twice as fast.

Using gold-standard POS tagging, the best classifier model (SVM) achieved LR of 87.2% and LP of 88.3%, as shown in Table 4. Both measures surpass the previously known best results on parsing using gold-standard tagging. We also tested the SVM model using data automatically tagged by our POS tagger, and it achieved LR of 78.1% and LP of 81.1% for sentences $\leq$ 40 words, as shown in Table 3.

4.2 Classifier Ensemble Experiments

Classifier ensemble by itself has been a fruitful research direction in machine learning in recent years. The basic idea in classifier ensemble is that combining multiple classifiers can often give significantly better results than any single classifier alone. We experimented with three different classifier ensemble strategies: classifier stacking, meta-classifier, and simple voting.

Using the SVM classifier's results as a baseline, we tested these approaches on the development set. In classifier stacking, we collect the outputs from Maxent, DTree and TiMBL, which are all trained on a separate dataset from the training set (section 400-650 of the Penn Chinese Treebank, smaller than the original training set). We use their classification output as features, in addition to the original feature set, to train a new SVM model on the original training set. We achieved LR of 90.3% and LP of 90.5% on the development set, a 3.4% and 2.6% improvement in LR and LP, respectively. When tested on the test set, we gained 1% improvement in F1 when gold-standard POS tagging is used. When tested with automatic tagging, we achieved a 0.5% improvement in F1. Using Bikel's significant tester with 10000 times random shuffle, the p-value for LR and LP are 0.008 and 0.457, respectively. The increase in recall is statistically significant, and it shows classifier stacking can improve performance.

On the other hand, we did not find meta-classification and simple voting very effective. In simple voting, we make the classifiers to vote in each step for every parse action. The F1 of simple voting method is downgraded by 5.9% relative to SVM model's F1. By analyzing the inter-agreement among classifiers, we found that there were no cases where Maxent's top output and DTree's output were both correct and SVM's output was wrong. Using the top output from Maxent and DTree directly does not seem to be complementary to SVM.

In the meta-classifier approach, we first collect the output from each classifier trained on sec-

MODEL	≤ 40 words				≤ 100 words				Unlimited			
	LR	LP	F1	POS	LR	LP	F1	POS	LR	LP	F1	POS
Bikel & Chiang 2000	76.8%	77.8%	77.3%	-	73.3%	74.6%	74.0%	-	-	-	-	-
Levy & Manning 2003	79.2%	78.4%	78.8%	-	-	-	-	-	-	-	-	-
Xiong et al. 2005	78.7%	80.1%	79.4%	-	-	-	-	-	-	-	-	-
Bikel's Thesis 2004	78.0%	81.2%	79.6%	-	74.4%	78.5%	76.4%	-	-	-	-	-
Chiang & Bikel 2002	78.8%	81.1%	79.9%	-	75.2%	78.0%	76.6%	-	-	-	-	-
Jiang's Thesis 2004	80.1%	82.0%	81.1%	92.4%	-	-	-	-	-	-	-	-
Sun & Jurafsky 2004	**85.5**%	**86.4**%	**85.9**%	-	-	-	-	-	**83.3**%	**82.2**%	**82.7**%	-
DTree model	71.8%	76.9%	74.4%	92.5%	69.2%	74.5%	71.9%	92.2%	68.7%	74.2%	71.5%	92.1%
SVM model	78.1%	81.1%	79.6%	92.5%	75.5%	**78.5**%	77.0%	92.2%	75.0%	78.0%	76.5%	92.1%
Stacked classifier model	79.2%	81.1%	80.1%	92.5%	**76.7**%	78.4%	**77.5**%	92.2%	76.2%	78.0%	77.1%	92.1%

Table 3: Comparison with related work on the test set using automatically generated POS

tion 1-210 (roughly 3/4 of the entire training set). Then specifically for Maxent, we collected the top output as well as its associated probability estimate. Then we used the outputs and probability estimate as features to train an SVM classifier that makes a decision on which classifier to pick. Meta-classifier results did not change at all from our baseline. In fact, the meta-classifier always picked SVM as its output. This agrees with our observation for the simple voting case.

5 Comparison with Related Work

Bikel and Chiang (2000) constructed two parsers using a lexicalized PCFG model that is based on Collins' model 2 (Collins, 1999), and a statistical Tree-adjoining Grammar(TAG) model. They used the same train/development/test split, and achieved LR/LP of 76.8%/77.8%. In Bikel's thesis (2004), the same Collins emulation model was used, but with tweaked head-finding rules. Also a POS tagger was used for assigning tags for unseen words. The refined model achieved LR/LP of 78.0%/81.2%. Chiang and Bikel (2002) used inside-outside unsupervised learning algorithm to augment the rules for finding heads, and achieved an improved LR/LP of 78.8%/81.1%. Levy and Manning (2003) used a factored model that combines an unlexicalized PCFG model with a dependency model. They achieved LR/LP of 79.2%/78.4% on a different test/development split. Xiong et al. (2005) used a similar model to the BBN's model in (Bikel and Chiang, 2000), and augmented the model by semantic categorical information and heuristic rules. They achieved LR/LP of 78.7%/80.1%. Hearne and Way (2004) used a Data-Oriented Parsing (DOP) approach that was optimized for top-down computation. They achieved F1 of 71.3 on a different test and training set. Jiang (2004) reported LR/LP of 80.1%/82.0% on sentences $\leq$ 40 words (results not available for sentences $\leq$ 100 words) by applying Collins' parser to Chinese. In Sun and Jurafsky (2004)'s work on Chinese shallow semantic parsing, they also applied Collin's parser to Chinese. They reported up-to-date the best parsing performance on Chinese Treebank. They achieved LR/LP of 85.5%/86.4% on sentences $\leq$ 40 words, and LR/LP of 83.3%/82.2% on sentences $\leq$ 100 words, far surpassing all other previously reported results. Luo (2003) and Fung et al. (2004) addressed the issue of Chinese text segmentation in their work by constructing character-based parsers. Luo integrated segmentation, POS tagging and parsing into one maximum-entropy framework. He achieved a F1 score of 81.4% in parsing. But the score was achieved using 90% of the 250K-CTB (roughly 2.5 times bigger than our training set) for training and 10% for testing. Fung et al.(2004) also took the maximum-entropy modeling approach, but augmented by transformation-based learning. They used the standard training and testing split. When tested with gold-standard segmentation, they achieved a F1 score of 79.56%, but POS-tagged words were treated as constituents in their evaluation.

In comparison with previous work, our parser's accuracy is very competitive. Compared to Jiang's work and Sun and Jurafsky's work, the classifier ensemble model of our parser is lagging behind by 1% and 5.8% in F1, respectively. But compared to all other works, our classifier stacking model gave better or equal results for all three measures. In particular, the classifier ensemble model and SVM model of our parser achieved second and third highest LP, LR and F1 for sentences $\leq$ 100 words as shown in Table 3. (Sun and Jurafsky did not report results on sentences $\leq$ 100 words, but it is worth noting that out of all the test sentences, only 2 sentences have length > 100).

Jiang (2004) and Bikel (2004)[3] also evaluated their parsers on the test set for sentences ≤ 40 words, using gold-standard POS tagged input. Our parser gives significantly better results as shown in Table 4. The implication of this result is two-fold. On one hand, it shows that if POS tagging accuracy can be increased, our parser is likely to benefit more than the other two models; on the other hand, it also indicates that our deterministic model is less resilient to POS errors. Further detailed analysis is called for, to study the extent to which POS tagging errors affects the deterministic parsing model.

Model	LR	LP	F1
Bikel's Thesis 2004	80.9%	84.5%	82.7%
Jiang's Thesis 2004	84.5%	88.0%	86.2%
DTree model	80.5%	83.9%	82.2%
Maxent model	81.4%	82.8%	82.1%
SVM model	87.2%	**88.3**%	87.8%
Stacked classifier model	**88.3**%	88.1%	**88.2**%

Table 4: Comparison with related work on the test set for sentence ≤ 40 words, using gold-standard POS

To measure efficiency, we ran two publicly available parsers (Levy and Manning's PCFG parser (2003) and Bikel's parser (2004)) on the standard test set and compared the runtime[4]. The runtime of these parsers are shown in *minute:second* format in Table 5. Our SVM model is more than 2 times faster than Levy and Manning's parser, and more than 13 times faster than Bikel's parser. Our DTree model is 40 times faster than Levy and Manning's parser, and 270 times faster than Bikel's parser. Another advantage of our parser is that it does not take as much memory as these other parsers do. In fact, none of the models except MBL takes more than 60 megabytes of memory at runtime. In comparison, Levy and Manning's PCFG parser requires more than 400 mega-bytes of memory when parsing long sentences (70 words or longer).

6 Discussion and future work

One unique attraction of this deterministic parsing framework is that advances in machine learning field can be directly applied to parsing, which opens up lots of possibilities for continuous improvements, both in terms of accuracy and efficiency. For example, in this paper we experimented with one method of simple voting. An alternative way of doing simple voting is to let the parsers vote on membership of constituents after each parser has produced its own parse tree (Henderson and Brill, 1999), instead of voting at each step during parsing.

Model	runtime
Bikel	54m 6s
Levy & Manning	8m 12s
Our DTree model	**0m 14s**
Our Maxent model	0m 24s
Our SVM model	3m 50s

Table 5: Comparison of parsing speed

Our initial attempt to increase the accuracy of the DTree model by applying boosting techniques did not yield satisfactory results. In our experiment, we implemented the AdaBoost.M1 (Freund and Schapire, 1996) algorithm using resampling to vary the training set distribution. Results showed AdaBoost suffered severe overfitting problems and hurts accuracy greatly, even with a small number of samples. One possible reason for this is that our sample space is very unbalanced across the different classes. A few classes have lots of training examples while a large number of classes are rare, which could raise the chance of overfitting.

In our experiments, SVM model gave better results than the Maxent model. But it is important to note that although the same set of features were used in both models, a degree 2 polynomial kernel was used in the SVM classifier while Maxent only has degree 1 features. In our future work, we will experiment with degree 2 features and L1 regularization in the Maxent model, which may give us closer performance to the SVM model with a much faster speed.

7 Conclusion

In this paper, we presented a novel deterministic parser for Chinese constituent parsing. Using gold-standard POS tags, our best model (using stacked classifiers) runs in linear time and has labeled recall and precision of 88.3% and 88.1%, respectively, surpassing the best published results. And with a trade-off of 5-6% in accuracy, our DTree and Maxent parsers run at speeds 40-270 times faster than state-of-the-art parsers. Our re-

[3]Bikel's parser used gold-standard POS tags for unseen words only. Also, the results are obtained from a parser trained on 250K-CTB, about 2.5 times bigger than CTB 1.0.

[4]All the experiments were conducted on a Pentium IV 2.4GHz machine with 2GB of RAM.

sults have shown that the deterministic parsing framework is a viable and effective approach to Chinese parsing. For future work, we will further improve the speed and accuracy of our models, and apply them to more Chinese and multilingual natural language applications that require high speed and accurate parsing.

Acknowledgment

This work was supported in part by ARDA's AQUAINT Program. We thank Eric Nyberg for his help during the final preparation of this paper.

References

Daniel M. Bikel and David Chiang. 2000. Two statistical parsing models applied to the Chinese Treebank. In *Proceedings of the Second Chinese Language Processing Workshop, ACL '00*.

Daniel M. Bikel. 2004. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models.* Ph.D. thesis, University of Pennsylvania.

Yuchang Cheng, Masayuki Asahara, and Yuji Matsumoto. 2004. Deterministic dependency structure analyzer for Chinese. In *Proceedings of IJCNLP '04*.

Yuchang Cheng, Masayuki Asahara, and Yuji Matsumoto. 2005. Machine learning-based dependency analyzer for Chinese. In *Proceedings of ICCC '05*.

David Chiang and Daniel M. Bikel. 2002. Recovering latent information in treebanks. In *Proceedings of COLING '02*.

Michael John Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing.* Ph.D. thesis, University of Pennsylvania.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2004. Timbl version 5.1 reference guide. Technical report, Tilburg University.

Yoav Freund and Robert E. Schapire. 1996. Experiments with a new boosting algorithm. In *Proceedings of ICML '96*.

Pascale Fung, Grace Ngai, Yongsheng Yang, and Benfeng Chen. 2004. A maximum-entropy Chinese parser augmented by transformation-based learning. *ACM Transactions on Asian Language Information Processing*, 3(2):159–168.

Mary Hearne and Andy Way. 2004. Data-oriented parsing and the Penn Chinese Treebank. In *Proceedings of IJCNLP '04*.

John Henderson and Eric Brill. 1999. Exploiting diversity in natural language processing: Combining parsers. In *Proceedings of EMNLP '99*.

Zhengping Jiang. 2004. Statistical Chinese parsing. Honours thesis, National University of Singapore.

Taku Kudo and Yuji Matsumoto. 2000. Use of support vector learning for chunk identification. In *Proceedings of CoNLL and LLL '00*.

Roger Levy and Christopher D. Manning. 2003. Is it harder to parse Chinese, or the Chinese Treebank? In *Proceedings of ACL '03*.

Xiaoqiang Luo. 2003. A maximum entropy Chinese character-based parser. In *Proceedings of EMNLP '03*.

David M. Magerman. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. thesis, Stanford University.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *Proceedings of EMNLP '04*.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING '04*.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the IWPT '05*.

Honglin Sun and Daniel Jurafsky. 2003. The effect of rhythm on structural disambiguation in Chinese. In *Proceedings of SIGHAN Workshop '03*.

Honglin Sun and Daniel Jurafsky. 2004. Shallow semantic parsing of Chinese. In *Proceedings of the HLT/NAACL '04*.

Antal van den Bosch and Jakub Zavrel. 2000. Unpacking multi-valued symbolic features and classes in memory-based language learning. In *Proceedings of ICML '00*.

Deyi Xiong, Shuanglong Li, Qun Liu, Shouxun Lin, and Yueliang Qian. 2005. Parsing the Penn Chinese Treebank with semantic knowledge. In *Proceedings of IJCNLP '05*.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT '03*.

Le Zhang, 2004. *Maximum Entropy Modeling Toolkit for Python and C++*. Reference Manual.