Improving Part-of-speech Tagging for Context-free Parsing

Xiao Chen and Chunyu Kit
Department of Chinese, Translation and Linguistics
City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong SAR, P. R. China
{cxiao2,ctckit}@cityu.edu.hk

Abstract

In this paper, we propose a factored parsing model consisting of a lexical and a constituent model. The discriminative lexical model allows the parser to utilize rich contextual features beyond those encoded in the context-free grammar (CFG) in use. Experiment results reveal that our parser achieves statistically significant improvement in both parsing and tagging accuracy on both English and Chinese.

1 Introduction

Part-of-speech (POS) tagging, or tagging for short, is usually considered a front-end preparation for parsing. Folk wisdom holds that accurate tagging result is helpful to alleviate the syntactic ambiguity problem in parsing, motivating a huge amount of research on perfecting tagging techniques.

In the past two decades, most POS tagging systems were based on a sequential classification approach, decomposing a sequence labeling task into a series of classification subtasks. The state of the art of tagging was achieved by virtue of well-developed machine learning methods, e.g. the Maximum Entropy model as in Ratnaparkhi (1996) and the Support Vector Machine as in Gimenez and Marquez (2003). All these techniques boosted the performance of POS tagging significantly. The error rate of the best English POS tagger is less than 3% (Ratnaparkhi, 1996; Toutanova et al., 2003; Shen et al., 2007), very close to the inter-annotator discrepancy of the Penn Treebank (Marcus et al., 1993).

For a long time, however, parsing seems to have been evolving in parallel to tagging without much interaction with each other. Generative parsers (Collins, 1997; Charniak, 2000; Charniak and Johnson, 2005; Petrov and Klein, 2007), owing to their generative nature, all include a lexical probability model in the form $P(w|t)$. The information used to predict the POS tag of a word mainly comes from the word itself and/or the ancestors that derive this word in a tree. Local context, which is proved to be the most useful information source for tagging (Ratnaparkhi, 1996; Toutanova and Manning, 2000), is not efficiently utilized by these parsers. Another noticable fact is that these high-performance parsers cannot do a better job of POS tagging than most of the state-of-the-art taggers (see Section 4 for a comparison). This is quite against our intuition that a parser having access to syntactic and contextual information from all over an input sentence should outperform a tagger that is limited to utilizing only local context and sequential dependency. This is an observation in Charniak et al. (1996). But their explanation is that a parser is built to find phrase markers, not tags. Then an interesting question arises: if a parser cannot tag words better, how can we expect it to do better on phrases?

Driven by this question, our research is intended to explore an approach to integrating parsing with the strength of successful tagging, in the hope of improving both. Firstly, we factor the conventional PCFG-based parsing into a lexical and a constituent model. Then, two candidate lexical models are formulated to incorporate enriched contextual features and sequential dependency into the parsing, with an averaged perceptron algorithm for parameter estimation. Our experiments on English and Chinese treebanks show that this approach achieves a significant performance enhancement in both parsing and tagging over the baseline Berkeley parser.

2 A Factored Parsing Model

2.1 POS Tagging in Parsing

For PCFG-based parsing, the joint probability $\mathcal{P}(T, S)$ for a parse tree T of a sentence S is usu-

Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 1260–1268,
Chiang Mai, Thailand, November 8 – 13, 2011.

ally defined as the product of the probability of every CFG rule $\mathcal{R}$ involved in T:

$$\mathcal{P}(T,S) = \prod_{\mathcal{R}\in T} \mathcal{P}(\mathcal{R})^{|\mathcal{R}|} \quad (1)$$

where $|\mathcal{R}|$ is the number of occurrences of $\mathcal{R}$ in T. In a CFG, such as the one induced from the Penn Treebank, there are usually two types of grammar rules: *lexical rule*, e.g. `NN` $\rightarrow$ `consumer`, to indicate a possible POS-tag for a word, and *constituent rule*, e.g. `PP` $\rightarrow$ `IN NP`, to indicate the composition of a constituent.

Let r denote a lexical rule and R a constituent rule, the probability defined in (1) can be rewritten as

$$\mathcal{P}(T,S) = \prod_{R\in T} \mathcal{P}(R)^{|R|} \prod_{r\in T} \mathcal{P}(r)^{|r|} \quad (2)$$

Accordingly, the parsing model can be decomposed into two factors:

$$\mathcal{P}(T,S) = \mathcal{P}(\mathcal{C},S)\mathcal{P}(\mathcal{L},S) \quad (3)$$

where $\mathcal{P}(\mathcal{C},S) = \prod_{R\in T} \mathcal{P}(R)^{|R|}$ is a constituent model and $\mathcal{P}(\mathcal{L},S) = \prod_{r\in T} \mathcal{P}(r)^{|r|}$ a lexical model. The best parse is then selected by the joint inference with these two submodels. This factored-out lexical model provides the flexibility of integrating various well-developed POS tagging techniques into parsing, and it is also easier for optimization, in contrast to a complex joint model. It is reasonable that a better lexical model is expected to have better effects on parsing.

2.2 Product of Experts

The two separated models may score in different magnitudes, even if they are all properly normalized. Usually, when combining two heterogeneous models, a weighting scheme is needed to balance their unequal effect. For this, we introduce another parameter λ to our factorized parsing model as

$$\mathcal{P}(T,S) = \mathcal{P}(\mathcal{L},S)^{\lambda}\mathcal{P}(\mathcal{C},S) \quad (4)$$

This is known as a product-of-experts (Hinton, 2002; Cohen and Smith, 2007), where a combined distribution is defined by multiplying several component distributions and renormalizing them. The parameter λ can be tuned with the Gold Section Search algorithm (Press et al., 2007) on a development set, using the F-measure of PARSEVAL (Black et al., 1991) as objective function for training.

3 Lexical Model

3.1 Model I

Sequential dependency and local context have shown their strength in tagging disambiguation (Ratnaparkhi, 1996; Toutanova and Manning, 2000; Toutanova et al., 2003). However, in the high-performance PCFG-based parsers (Collins, 1997; Charniak, 2000; Petrov and Klein, 2007), none of these features can be used due to the generative nature of these parsing models.

To address this problem and incorporate the advantages of POS tagging technique into parsing, we propose a discriminative lexicon model following the global linear model (Collins, 2002). Let $\tau = \{t_0, t_1, t_2 ... t_n\}$ be a tag sequence for a sentence $S = \{w_0, w_1, w_2 ... w_n\}$, we define the score of a tagged sequence to be

$$\text{SCORE}(\tau, S) = \theta \cdot \mathbf{f}(\tau, S) = \sum_{i=1}^{n} \theta \cdot f(\tau_i, S) \quad (5)$$

where $\mathbf{f}(\tau, S)$ is a global feature vector and θ a vector of associated weights. A global feature is defined through a collection of local features $f(\tau_i, S)$. We train θ on a treebank using an averaged perceptron algorithm similar to the one presented in Collins and Duffy (2002) and Collins (2002). The number of iterations needed is optimized on the development set.

However, introducing sequential dependency into this lexical model would cause a severe efficiency problem with the joint inference for parsing. When the Viterbi algorithm is used to search for the best parse tree, its efficiency relies heavily on tree structure. The interdependency of adjacent POS tags actually changes the structure of the parse under operation. For example, to determine the best sub-tree for a non-terminal `XP` covering a span of words $[w_i ... w_j]$, we need to calculate the score for w_i's tag, which may largely depend on w_{i-1}'s tag. Unfortunately, such information is not available from the current word span under processing. The inference algorithm thus has to keep all possible sub-trees in memory, resulting in an intractable inference problem.

To deal with this joint inference problem, we approximate the lexical model in a unigram-factored form:

$$\mathcal{P}(\mathcal{L},S) = \mathcal{P}(\mathcal{L}|S)\mathcal{P}(S) \propto \prod_{i\in[0,n]} \mathcal{P}(t_i|S) \quad (6)$$

where $\mathcal{P}(S)$ is the probability of a given input sentence, a constant that can be dropped if search only for the best parse, and t_i a candidate tag for w_i. The conditional distribution $\mathcal{P}(t_i|S)$ can be estimated by

$$\mathcal{P}(t_i|S) = \frac{\sum_{\tau \in \mathbb{T}(t_i)} \exp\left(\text{SCORE}(\tau, S)\right)}{\sum_{\tau' \in \mathbb{T}} \exp\left(\text{SCORE}(\tau', S)\right)} \quad (7)$$

where $\mathbb{T}$ is the set of all possible tag sequences for S, and $\mathbb{T}(t_i)$ a subset of $\mathbb{T}$, consisting of all tag sequences that contain t_i. The numerator is the sum of scores for all possible tag sequences with t_i for w_i. The denominator is the total score of all possible tag sequences of the input sentence. For efficient calculation, we also adopt variants of the forward and backward algorithm, which are similar to those for HMM (Baum and Eagon, 1967; Baum and Sell, 1968). Different from the conventional tagging systems, our lexical model does not generate the best tag sequence for the whole sentence, but a lattice of tags, on which the joint inference with the constituent model can be performed.

In our model, feature functions $f_m(\tau_i, S)$ are primarily binary, each of which maps the local context to 1 if its feature exists, or to 0 otherwise. The only exception is the first one. Using a similar design as in discriminative re-scoring parsing models (Collins, 2000; Charniak and Johnson, 2005; Huang, 2008), we have the first feature f_0 to be the logarithm of the probability of t_i being w_i's tag in S, as proposed in Petrov and Klein (2007), which is $f_0(\tau_i, S) = \log q(t_i, S)$.[1]

Beside $f_0(\tau_i, S)$, all other features are similar to those in the previous works (Ratnaparkhi, 1996; Toutanova et al., 2003; Shen et al., 2007), as listed in Table 1.

The signature features *prefix* and *suffix*, as used in Toutanova et al. (2003), each return a substring of certain length from the current word. In our experiments, this length ranges from 1 to 8. Since these prefix and suffix features are blindly extracted with templates that can be applied to any language, our lexical model is more language-independent than those using a predefined language-specific affix list.

Type	Feature Template	
Word	$[w_i]\ (i = -2, -1, 0, 1, 2)$	$\&[t_0]$
	$[w_{i-2}, w_{i-1}]\ (i = 0, 1, 2)$	$\&[t_0]$
	$[w_{-1}, w_1]$	$\&[t_0]$
	$[w_{i-2}, w_{i-1}, w_i]\ (i = 0, 1, 2)$	$\&[t_0]$
Tag	$[t_i]\ (i = -2, -1, 1, 2)$	$\&[t_0]$
	$[t_{i-2}, t_{i-1}]\ (i = 0, 3)$	$\&[t_0]$
Signature	punctuator	$\&[t_0]$
	digit	$\&[t_0]$
	prefix	$\&[t_0]$
	suffix	$\&[t_0]$

Table 1: Feature templates of lexical model

For feature selection, we also follow other researchers' previous works to use a simply cut-off threshold. During the process to generate features for each word in the training treebank, a feature is not included in the final model if its count falls below a predefined threshold. We set the threshold to 2 for word and tag features, and to 35 for signature features.

3.2 Model II

The lexical model I enables the parsing to utilize a large variety of features other than those encoded in the CFG in use. However, the computation of forward/backward variables is expensive in both time and space. This inefficiency is practically more severe in training, which requires repetitive computation of these values for many times. Lexical model II is proposed as an aggressive approximation of model I, aimed at improving its efficiency at the least cost of performance.

The inefficiency of model I is primarily due to the complexity of the forward/backward procedure. For model II, we propose a lazy procedure as a simple approximation, which calculates the conditional probability $\mathcal{P}'(t_i|S)$ deterministically from left to right. As the procedure proceeds, all features that contain a preceding tag are instantiated with the best predicted tags for those previously processed words. For the features that contain a succeeding tag, we use the tag with the highest score by $f_0(\tau_i, S)$. This lazy procedure traces only one arbitrary best tag sequence. Its time complexity is linear in the length of input sentence.

Accordingly, for lexical model II, the distribu-

[1] In the lexical model, for $t_i = \mathcal{A}$ (a non-terminal symbol), the score $q(t_i, S)$ is calculated by:

$$q(t_i, S) = \frac{\sum_x O(\mathcal{A}_x, i, i+1)\mathcal{P}(\mathcal{A}_x \rightarrow w_i)}{I(ROOT, 0, |S|)} \quad (8)$$

where x is the latent variable (Petrov et al., 2006), and $I(\cdot)$ and $O(\cdot)$ are *inside* and *outside* scores. For more details of $q(t_i, S)$, please refer to Figure 3 in Petrov and Klein (2007).

tion $\mathcal{P}'(t_i|S)$ is estimated as

$$\mathcal{P}'(t_i|S) \propto \frac{\exp\left(\theta \cdot \mathbf{f}(\tau_i^*, S)\right)}{\sum\limits_{\substack{j\in[0,n-1],\\ t_j\in E(S)}} \exp\left(\theta \cdot \mathbf{f}(\tau_j^*, S)\right)} \quad (9)$$

where τ_i^* is the arbitrary best tag sequence and $E(S)$ the collection of all possible POS tags for every word in S. The denominator is introduced for two reasons. Firstly, it is used to map the score of every t_i to the interval $[0, 1]$, so as to make the lexical model to have the similar magnitude as the constituent model. Secondly, we deliberately let it not to be a local normalizer. This allows some confident tags to vote stronger than others in the joint inference.

4 Experiments

We implement a parser for experiments with necessary support from the open source Berkeley parser. To evaluate the performance of this parsing model across languages, we conduct a number of experiments on both English and Chinese treebanks, using the WSJ section of Penn Treebank 3.0 (LDC99T42) and the Chinese Treebank 5.1 (LDC2005T01U01). Since both the constituent and lexical models use the PCFG-LA trained with the Berkeley parser,[2] we take it as the baseline for our evaluation. For comparison purpose, we use exactly the same splits of the treebanks as in Petrov and Klein (2007), as listed in Table 2.

	English	Chinese
Train.	Section 2-21	Art. 1-270,400-1151
Dev.	Section 22	Art. 301-325
Test.	Section 23	Art. 271-300

Table 2: Experiment Setup

As mentioned in Section 3, the parameter estimation for the lexical model requires reparsing the training treebank in use for calculating $q(t_i, S)$ with the PCFG-LA trained with the Berkeley parser. In order to obtain a representative set of training examples, the PCFG-LA is expected to create as much noise as it does in testing. For this, the training set for each language is divided into 20 folds as in Collins (2000) and Charniak and Johnson (2005). Each fold is reparsed with the PCFG-LA trained on the other 19 folds. The number of split-and-merger iterations is set to 6 for the training of all these grammars on the treebanks of both languages.

To examine how much the rich contextual features introduced into the lexical model improve parsing performance, the tagging accuracy of our parser is compared with the state-of-the-art taggers, e.g. the open source Stanford tagger. Although this tagger was developed several years ago, its performance on English is still in the lead according to Spoustová et al. (2009). We train and test this tagger on the same datasets as for parsing, using the default parameter settings for each language.

All parsing results are evaluated with the standard `evalb`. It is noteworthy that the tags are not counted as part of parsing output. The tagging accuracy has to be evaluated with our own program, for `evalb` eliminates some `DELETE LABEL`s when evaluating tagging.

4.1 Tagging Helps Parsing

Table 3 compares the performance of our parser and other baseline systems. When trained on the same data set, the Stanford tagger indeed outperforms the Berkeley parser and the re-scoring parser[3] on POS tagging for both languages. Then, we impose various tagging results into parsing to see how they affect parsing performance. Both the gold POS tags and the output of the Stanford tagger are submitted to the Berkeley parser, using the "`-goldPOS`" option. The results are presented in Table 3, denoted as *Berkeley+GoldTag* and *Berkeley+Stanford* respectively.[4] Surprisingly, however a significant loss of parsing performance is caused by imposing the better yet non-perfect tagging output of the Stanford tagger onto the Berkeley parser. We can see that better tagging does not necessarily improve parsing if the two separate systems work in the conventional "tag then parse" order.

The two lexical models are tested in the other experiments. We use all features as listed in Table

[2]The version released in September, 2009. The option "-Chinese" is not used for parsing Chinese, for it does not give the best parsing performance. All the PCFG-LA mentioned in this paper are trained with this version of the Berkeley parser with default settings.

[3]C&J Parser: the open source re-scoring parser of Charniak and Johnson (2005)

[4]Note that the tagging accuracy of *Berkeley+GoldTag* is not 100%. When some of the gold tags could cause parsing failure, the Berkeley parser will skip them and use its own tagging output.

		English			Chinese		
		Parsing	Tagging	Lex.	Parsing	Tagging	Lex.
		F1 (%)	Accuracy (%)		F1 (%)	Accuracy (%)	
	Stanford	-	97.47	-	-	95.44	-
	C&J Parser	91.40	94.50	-	-	-	-
	Berkeley	89.87	97.30	-	83.22	95.33	-
	Berkeley+Stanford	89.13	97.44	-	81.35	95.44	-
	Berkeley+GoldTag	89.88	99.82	-	88.35	99.75	-
LM I	Word	89.95	97.43	97.18	84.37	96.38	95.82
	Word+Tag	**90.02**	97.46	97.41	84.48	96.57	96.27
	Word+Tag+λ	89.99	97.52	97.41	**84.59**	96.64	96.27
	Word+Tag w/o f_0	89.98	97.51	97.18	84.09	95.95	94.48
LM II	Word	89.92	97.41	97.28	84.18	96.09	95.79
	Word+Tag	90.01	97.49	97.39	84.27	96.34	96.04
	Word+Tag+λ	89.99	97.59	97.39	84.44	96.34	96.04

Table 3: Results on English and Chinese testing sets (all sentences).

1 for English. For Chinese, however, we select a smaller subset of it by removing word trigram templates and all tag templates that contain any tag to the right of the current word.[5] We find that using such tags as features cause a loss of performance, which is consistent with the default setting of the Stanford tagger for Chinese. Another difference in feature design for these two languages is the length of prefix and suffix. The max length of affix is set to 6 for English and 3 for Chinese. This is because the average word length is different in the two languages. And we use no specific Chinese characters for the feature templates of *punctuator* and *digit*, in order to maintain the language independency of our models as much as possible.

As presented in Table 3, our parser with both candidate lexical models performs significantly better than the baseline Berkeley parser.[6] For Chinese, the best labelled F-score 84.59% is 1.3% beyond the baseline 83.22%. For English, our best result 90.02% is even better than 89.88% using gold POS tags. It is evident that our enhanced lexical model can effectively help parsing. With its help, our parser also outperforms both the Berkeley parser and the Stanford tagger on tagging. The observation that the improvement on English looks not as much as that on Chinese could be explained by the fact that the tagging accuracy of English is very close to the inter-annotator discrepancy of the Penn Treebank, leaving a too tiny room for further improvement. According to Dalrymple (2006), parsing ambiguity in about 30% sentences cannot be reduced even by a perfect tagger, implying that improving tagging may only have a limited influence on parsing.

We also examine the effect of different feature set in both lexical models. With only word features, the parser already achieves some improvement on both parsing and tagging, in consistence with the findings in Toutanova et al. (2003). The surrounding words provide most information for the disambiguation for tagging. However, the weight parameter λ seems not as effective as expected for English. This might be caused by the overfitting when it is tuned it on the development set. We also try to remove the feature f_0 from the model (w/o f_0). Then, the performance is still higher than the baseline Berkeley parser, indicating that the effectiveness of our lexical model does not rely on this feature too much. More interestingly, the approximate deterministic lexical model II achieves a similar performance as the complex lexical model I. In practice, model II is a better choice with a lower computational complexity.

Before the joint inference with the constituent model, a lexical model calculates a score for every candidate tag of each word. A choice is to collect the tags with the highest scores as a baseline output of tagging. The tagging accuracy of a lexical model can be compared with this output, as shown in the last column "Lex." for each language in Table 3. Comparing this column with the tagging accuracy of our parser in the second column for each language, we can observe an increase by 0.1-0.3%, indicating that syntactic knowledge as used in our parser does help the disambiguation for tagging.

[5]Specifically, tag templates with $i > 0$ are removed for experiments on Chinese.

[6]All results are tested with Dan Bikel's Randomized Parsing Evaluation Comparator (http://www.cis.upenn.edu/~dbikel/download/compare.pl), resulting in $p < 0.05$.

System	F1 (%)
Berkeley	89.87
Charniak (2000)	89.70
Collins (2000)	89.70
Bod (2003)	90.70
Henderson (2004)	90.10
Charniak and Johnson (updated 2006)	91.40
Huang (2008)	91.69
Attia et al. (2010)	89.88
This work	90.02

Table 4: Comparison with other parsers on English testing set

System	F1 (%)
Berkeley	83.22
Burkett and Klein (2008)	84.24
This work	84.59

Table 5: Comparison with other parsers on Chinese testing set

4.2 Comparison with Previous Work

Table 4 and 5 compare the performance of our parser with other high-performance parsers. Those parsers using self-training or parser combination methods are not included in this comparison, because they use extra resources or more than one parsing model.

For English, our parser outperforms all generative parsers (Collins, 1997; Charniak, 2000). But there is still a gap to the discriminative re-scoring methods (Charniak and Johnson, 2005; Huang, 2008). Given that our parser only improves the lexical model, the re-scoring method using a variety of subtree features indeed has some advantages. For Chinese, our parser has achieved the best performance so far on this data set. It is even better than the one that employs additional knowledge from the Chinese-English Parallel Treebank (Burkett and Klein, 2008).

5 Analysis

The experiments reported in the previous section seems to give a positive answer to the question: whether tagging can help parsing. However, several other questions follow: How does tagging affect paring? Why can this independently optimized lexical model improve parsing? We will look into these issuess in this section.

5.1 Profiling the Improvement of Tagging

We firstly divide the word/tag pairs in the testing data into to two groups: known v.s. unknown, according to the definition in Jin and Chen (2009). The tagging performance of our parser and the two baseline systems are compared in Table 6.

	English		Chinese	
	Known	Unk.	Known	Unk.
Stanford	98.09	77.33	96.47	72.02
Berkeley	97.93	76.13	96.91	59.23
This work	98.09	80.04	97.59	73.81

Table 6: Tagging accuracy (%) for known and unknown word/tag pairs

It is evident that our parser outperforms the Berkeley parser on both groups. The contrast on unknown tags is more significant, especially for Chinese. Since our parser uses the same constituent model as the Berkeley parser, this difference has to be explained by a better lexical model. A similar case can also be observed when our parser is compared against the Stanford tagger. Since our lexical model uses a very similar feature set as the Stanford tagger, the difference in performance should be attributed to the constituent model, which provides more detailed contextual information from the whole sentence to facilitate guessing unknown POS tags.

5.2 Profiling the Improvement of Parsing

In order to profile the parsing errors that can be fixed by better tagging, we count the number of correct constituents in the output of our parser and the other systems for a comparison under a variety of partition schemes, as shown in Table 7 and 8. The span length of a constituent is the number of words in it. Usually, a longer span correlates with a higher position near the root in a parse tree.

Table 7 shows that correct constituents increase in number in almost all types. Without any feature designed for a specific POS tag (such as `IN`), our lexical model brings about a general improvement to the parsing of both languages. Interestingly, however, the comparison in Table 8 presents a different view. The improvement mainly takes place in the constituents of span length ≤ 4, especially those base-level ones of length 1, which are parents of tags, e.g. `NP` $\rightarrow$ `NNS` and `VP` $\rightarrow$ `VBG`. The recognition of these constituents is greatly influenced by a lexical model. This influence tapers off on higher constituents, for the constituent model becomes more dominant when working on larger subtrees. This is also observed on the Chinese side. But the Chinese treebank has many different characteristics from English. According to our

English				Chinese			
Label	Berk.	E-LMII	Diff.	Label	Berk.	C-LMI	Diff.
NP	17049	17077	28	VP	1593	1638	45
VP	7953	7964	11	NP	3049	3073	24
ADJP	612	619	7	ADVP	301	313	12
PP	4748	4754	6	IP	780	795	15
S	5103	5109	6	PP	294	301	7
SBAR	1511	1514	3	CP	172	178	6
PRT	131	133	2	QP	181	185	4

Table 7: Correct constituents by constituent label

English						Chinese				
Span	Gold	C&J.	Berk.	E-LMII	Diff.	Span	Gold	Berk.	C-LMI	Diff.
1	6543	6034	5995	6015	20	1	2951	2594	2645	51
2	7645	7186	7159	7173	14	2	1385	1210	1221	11
3	5633	5237	5181	5188	7	3	844	679	699	20
4	3509	3183	3172	3175	3	4	528	433	452	19
≥ 5	20852	18385	17947	17970	23	≥ 5	2819	2234	2257	23

Table 8: Correct constituents by span length

statistics, about 34% of the constituents in the Chinese testing data are base-level ones. This explains why the impact of tagging to parsing for Chinese is much stronger than that for English. Since there are other languages that share similar characteristics with Chinese, our method should be also helpful in parsing these languages.

Currently, tagging and parsing are treated equally in a unified framework of parsing. However, our experiments show that our independently defined and optimized lexical model performs better than the one integrated into the Berkeley parser, especially for the recognition of the base-level constituents mentioned above. This is mostly attributable to the nature of a generative parsing model, for its lexical model can only use the features encoded in the grammar in use, which is never enough for accurate parsing. In fact, the research on POS tagging and chunking shows that the most important information for disambiguation at this level comes from local context (Toutanova et al., 2003), especially the surrounding words. Our current work is an attempt to fill this gap between tagging and parsing, by the way of enabling a parser to use rich contextual features in its lexical model.

6 Related Work

Most successful parsing models are generative models. Therefore, a large portion of previous related work did not change the generative nature of the lexical models involved (Goldberg et al., 2009; Huang and Harper, 2009; Attia et al., 2010). The key idea of these approaches is basically to advance smoothing techniques for the distribution $\mathcal{P}(w|t)$. Goldberg et al. (2009) adopt a trigram HMM tagging model trained on unannotated data to help prediting tags of rare words, but only the emission probabilities are used in parsing. Huang and Harper (2009) propose a better way to smooth the lexical model in a PCFG-LA parser similar to the Berkeley parser. Some morphological features are also used to handle unknown words in Chinese. In their evaluation, however, they exclude all unary rules that cause one of the major difficulties in parsing this language, according to our experiments. Their experiment settings obstruct comparing their results with others'. Attia et al. (2010) head in the same direction extending this method to other languages. Their experiment results are compared with ours in Table 4. The comparison shows that our method is more sophisticated in utilizing the sequential dependency between tags.

A motivation of our current work is that a discriminative lexical model can incorporate rich contextual features. In this respect, Cohen and Smith (2007) combine the strengths of a generative parsing model and a discriminative segmentation-tagging model under the notion of product-of-experts. Here we apply a similar idea to combine parsing and tagging. The discriminative re-scoring method (Charniak and Johnson, 2005; Huang, 2008) is also very successful. It enables a parser to use a large variety of local

and non-local features, so as to boost the performance. Interestingly, however, the comparison in Table 3 shows that the parser of Charniak and Johnson (2005) achieves the highest parsing performance with the lowest tagging accuracy. This parser seems to select the parse trees that contain correct constituents with relatively poor POS tags. This controversial result raises at least two questions: (a) Why correct constituent structures can be built on incorrect POS tags? (b) Does tagging have any effect on parsing? The first question points to the too strong generative capacity of the CFG in use. Note that without semantic and/or pragmatic knowledge as constraints, a CFG induced from a treebank usually can generate many possible but implausible structures for an input sentence. For the second question, the controversial result indeed gives us an impression that tagging has no effect on parsing, strongly against our intuition. Since tagging and parsing are unified into the same framework as a parser, it is reasonable to expect the correct tag sequence for a sentence to appear in the theoretically best parse tree. If a parser has to sacrifice tagging for parsing, it is suspectable that the re-scoring does not help the parser to reach its maximum capacity yet. Besides, the re-scoring method should have subsumed our current work. In fact, incorporating the sequential dependency between adjacent tags into the forest re-scoring model will also cause the efficiency problem in the inference as mentioned in Section 3. Given that no feature for tagging has been adopted in the forest re-scoring method so far, our current work is certainly a complement to this part.

The recent work of Rush et al. (2010) integrates parsing and tagging under the framework of dual-decomposition. This approach has the advantage to combine heterogeneous models, and solves the complex combinatory optimization problem via Lagrangian relaxation. However, it has a inevitable defect of inefficiency, since it requires to parse and tag the input sentence repetitively (usually 10 times for each sentence). Comparatively, our approach is more efficient. Note that lexical model II takes only a linear time in the length of input sentence. Without systematic comparison, however, it is difficult to tell which approach can provide a better performance. We will keep this for our future research.

7 Conclusion

In this work, we have proposed a parsing model which is factored into a lexical and a constituent model, in the hope of enabling the beneficial interaction between tagging and parsing. The relatively independent discriminative lexical model allows our parser to incorporate rich contextual features and even sequential dependency. Our experiments show that our lexical models can help parsing. With access to the syntactic knowledge from all over an input sentence, this parser outperforms the state-of-the-art POS tagging system in terms of tagging accuracy. Moreover, tagging should be an organic part of a parsing model to bring in a mutual positive effect for both parsing and tagging through joint inference. The conventional notion of tagging-parsing pipeline seems to leave no room for this possibility of enhancement.

Finally, individual words in an input sentence are found to be very useful in the disambiguation for POS tagging and even for recognizing base-level constituents. However, the structural parameterization of conventional parsing models cannot incorporate and utilize them effectively. Nevertheless, how to make use of all sorts of information available to enhance parsing is still a challenging research topic.

Acknowledgement

The research described in this paper was partially supported by the Research Grants Council (RGC) of HKSAR, China, through the GRF grant 9041597.

References

Mohammed Attia, Jennifer Foster, Deirdre Hogan, Joseph Le Roux, Lamia Tounsi, and Josef van Genabith. 2010. Handling unknown words in statistical latent-variable parsing models for Arabic, English and French. In *NAACL HLT 2010*, pages 67–75.

Leonard E. Baum and J. A. Eagon. 1967. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bull. Amer. Math. Soc.*, 73(3):360–363.

Leonard E. Baum and George R. Sell. 1968. Growth transformations for functions on manifolds. *Pacific J. Math.*, 27(2):211–227.

E. Black, S. Abney, D. Flickenger, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. San-

torini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of DARPA Speech and Natural Language Workshop*, pages 306–311.

Rens Bod. 2003. An efficient implementation of a new dop model. In *EACL 2003*, pages 19–26.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *EMNLP 2008*, pages 877–886.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *ACL 2005*, pages 173–180.

Eugene Charniak, Glenn Carroll, John Adcock, Anthony Cassandra, Yoshihiko Gotoh, Jeremy Katz, Michael Littman, and John McCann. 1996. Taggers for parsers. *Artif. Intell.*, 85:45–57.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *NAACL 2000*, pages 132–139.

Shay B. Cohen and Noah A. Smith. 2007. Joint morphological and syntactic disambiguation. In *EMNLP-CoNLL 2007*, pages 208–217.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. In *ACL 2002*, pages 263–270.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *ACL 1997*, pages 16–23.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *ICML 2000*, pages 175–182.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP 2002*, pages 1–8.

Mary Dalrymple. 2006. How much can part-of-speech tagging help parsing? *Nat. Lang. Eng.*, 12(4):373–389.

Jesus Gimenez and Lluis Marquez. 2003. Fast and accurate part-of-speech tagging: The SVM approach revisited. In *Proceedings of the Fourth RANLP*.

Yoav Goldberg, Reut Tsarfaty, Meni Adler, and Michael Elhadad. 2009. Enhancing unlexicalized parsing performance using a wide coverage lexicon, fuzzy tag-set mapping, and EM-HMM-based lexical probabilities. In *EACL 2009*, pages 327–335.

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *ACL 2004*, pages 95–102.

Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Zhongqiang Huang and Mary Harper. 2009. Self-training PCFG grammars with latent annotations across languages. In *EMNLP 2009*, pages 832–841.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL-HLT 2008*, pages 586–594.

Guangjin Jin and Xiao Chen. 2009. The fourth international chinese language processing bakeoff: Chinese word segmentation, named entity recognition and Chinese POS tagging. In *IJCNLP 2009*, pages 69–81.

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

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL 2007*, pages 404–411.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *COLING-ACL 2006*, pages 433–440.

William Press, Saul Teukolsky, William Vetterling, and Brian Flannery. 2007. *Numerical Recipes: the art of scientific computing*. Cambridge University Press, 3rd edition.

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *EMNLP 1996*, pages 133–142.

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *EMNLP 2010*, pages 1–11.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *ACL 2007*, pages 760–767.

Drahomíra "johanka" Spoustová, Jan Hajič, Jan Raab, and Miroslav Spousta. 2009. Semi-supervised training for the averaged perceptron POS tagger. In *EACL 2009*, pages 763–771.

Kristina Toutanova and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *SIGDAT EMNLP 2000*, pages 63–70.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL 2003*, pages 173–180.