

Chinese Statistical Parsing

Mary Harper^{†‡} and Zhongqiang Huang[†]

[†]Laboratory for Computational Linguistics and Information Processing
Institute for Advanced Computer Studies
University of Maryland, College Park
[‡]Human Language Technology Center of Excellence
Johns Hopkins University
{zqhuang, mharper}@umiacs.umd.edu

Abstract

This chapter describes several issues that are fundamental to achieving accurate Chinese parsing given available Chinese resources and the challenges of the Gale processing pipeline. For Gale, our parsing algorithm is expected to accurately parse various different materials, ranging from newswire text, which tends to be grammatically well formed, to n -best ASR outputs, many of which are poorly formed sentences. To address this challenge, we have re-implemented and enhanced the Berkeley parser to handle unknown Chinese words efficiently, parse difficult sentences robustly, and operate more efficiently. We also address issues related to training the parser for several different genres given a limited number of available training trees, the importance of matching word segmentation to the treebank segmentation standard to support accurate parsing, and the need for standardized tokenization for managing the types of things that will appear as input to the parser. Understanding and handling these issues is a prerequisite for achieving adequate parsing performance levels. We also investigate self-training with automatically labeled in-domain data to enhance parsing performance given the limited number of trees in the Chinese treebanks.

1 Introduction

There have been several attempts to develop high quality parsers for Chinese (Bikel and Chiang, 2000; Levy and Manning, 2003; Petrov and Klein, 2007b), but the state-of-the-art performance, around 83% F measure on Penn Chinese Treebank, achieved by

the Berkeley parser (Petrov et al., 2006; Petrov and Klein, 2007b) falls far short when compared to English¹. As pointed out in (Levy and Manning, 2003), there are many linguistic differences between Chinese and English, as well as structural differences between their corresponding treebanks, and some of these make it a harder task to parse Chinese. Additionally, the fact that the available treebanked Chinese materials are far more limited than for English also increases the difficulty of building high quality Chinese parsers. While there have been some investigations using treebanks to determine what makes Chinese hard to parse, here we attempt to learn about and control the factors that challenge Chinese parsers as applied to materials that are less controlled than a treebank.

Research that involves only parsing treebank test sets in some sense masks the challenges of providing high quality syntactic analyses for Chinese text and speech. First, Chinese documents are not segmented into words that conform to the word segmentation standard used by the Chinese treebank. Second, the Chinese treebanks contain punctuation, letters, and digits that are typically full-width; whereas, it is not uncommon for materials to contain mixtures of full- and half-width representations. It is in fact unlikely that the treebank would cover the combination of possible character types that may be observed in Chinese documents. Third, the treebank data is quite limited compared to the resources available for En-

¹In some cases, the parse results do not take into account the brackets of gold trees for sentences that do not produce a final parse in the score, and so the reported results are more optimistic than they should be.

glish parsing of newswire and speech genre, and so we must develop strategies to effectively parse materials in the text and speech genre covered by the GALE program given the resources at hand: Chinese Treebank 6 (CTB6) with parsed newswire and broadcast news sentences and Chinese Broadcast Conversation Treebank (BCTB) with parsed broadcast conversation sentences.

We will focus our parsing investigations in this chapter on the Chinese newswire and broadcast conversation genres because of the availability of treebanks representative of these genres. Unfortunately, the broadcast news treebank does not fully capture the spoken aspects it should represent in that it contains symbolic expressions that can be verbalized in a variety of ways. Although we can map this treebank to a verbalized form, this mapping is unfortunately less variable than an exact transcription of the corresponding speech would be. To maximize parsing performance, we process the treebanks differently for the newswire and broadcast conversation genres and train the models to match genre conditions. We evaluate genre matched parsers on a test set drawn from the respective newswire and broadcast conversation treebank.

The chapter is organized as follows. We describe the experimental data used in our investigation of Chinese parsing in Section 2 and the SParseval parse scoring tool used to measure parser performance in Section 3. Then we present the parser used in our study in Section 4. We discuss several factors that affect parsing performance, i.e., word segmentation in Section 5, text tokenization in Section 6, and mapping training trees from other genres to support newswire and broadcast conversation modeling in Section 7. We finally present results from using semi-supervised self-training to improve parsing performance in Section 8. The last section concludes this chapter.

2 Experimental Data

We used the Penn Chinese Treebank 6.0 (CTB6) and the Chinese Broadcast Conversation Treebank (BCTB) in our parsing studies, as well as a set of unlabeled sentences to support the use of semi-supervised self-training.

CTB6 contains 28k parsed sentences, including

news articles from Xinhua news agency (China-Mainland), Information Services Department of HKSAR (Hong Kong), and Sinorama magazine (Taiwan), as well as broadcast news from ACE evaluation (which we call CTB6BN). The news articles from the first three sources, with a total of 19k sentences, constitute the former Penn Chinese Treebank 5 (CTB5) and are the primary source of labeled data used for newswire experiments. Since the CTB5 corpus was collected during different time periods from different sources with a diversity of topics, in order to obtain a representative split of train-test-development sets, we divide it into blocks of 10 files in sorted order and for each block use the first file for development, the second for test, and the remaining for training. Although CTB6BN consists of parses for broadcast news transcriptions, it exhibits many of the characteristics of newswire text (it contains many nonverbal expressions, e.g., numbers and symbols, and is fully punctuated). Because of this similarity, we evaluate using this corpus as additional labeled training data for training the newswire model in Section 7.

BCTB contains 11k parsed sentences from three broadcast conversation sources (i.e., CCTV, CNN with Chinese translations, and Phoenix TV). In contrast to the CTB6BN data, the words in this treebank are fully verbal; however, most of the sentences in the treebank are punctuated. We manually selected the train-test-development split of the files to balance the number of sentences in each set. Since BCTB may be too small to train an accurate grammar for parsing speech, we also augment the training set with a verbalized version of CTB6, as discussed in Section 7.

In addition to the above two labeled treebanks, we also utilize a greater number of unlabeled sentences for investigating the use of semi-supervised self-training in Section 8. For the newswire experiments, 210k unlabeled sentences are selected from three newswire sources covering materials from China Mainland, Taiwan, and Hong Kong. For broadcast conversation experiments, 210k unlabeled sentences are selected from transcribed broadcast conversations.

We have developed a set of scripts to clean up the treebank trees used in our investigations. We remove comments and then delete empty nodes and

Data Split				
Genre	Train	Dev	Test	Unlabeled
NW	CTB5 14925 (~405k)	CTB5 1904 (~51k)	CTB5 1975 (~52k)	210k (~6255k)
	CTB5+CTB6BN 24416 (~679k)			
BC	BC 8149 (~110k)	BC 1297 (~14k)	BC 1497 (~17k)	210k (~2128k)
	CTB5+CTB6BN 24412 (~584k)			
	BC+CTB5+CTB6BN 32561 (~695k)			

Table 1: The number of sentences and words (in parentheses) of the data splits used in our experiments

nonterminal-yield unary rules, e.g., $NP \rightarrow VP$, using *tsurgeon* (Levy and Andrew, 2006). As nonterminal-yield unary rules are less likely to be posited by a statistical parser, it is common for parsers trained on the standard Chinese treebank to have substantially higher precision than recall. This gap between bracket recall and precision is alleviated without loss of parse accuracy by deleting the nonterminal-yield unary rules.

3 Sparseval Scoring

The *Sparseval* tool (Harper et al., 2005; Roark et al., 2006), which is implemented in C, was originally designed to support English speech-based bracket and head dependency scoring (recall, precision, and F-score) at the level of a demarcated chunk of speech such as a conversation side, while also supporting more traditional text-based scoring methods that require the input to the parser to match perfectly in words and sentence segments to the gold standard. The tool was developed to address the fact that output from an automatic speech recognition system is likely to contain word errors and the automatic segmentation of these words into sentences is likely to differ from those in the gold standard parses (Harper et al., 2005). To calculate the bracket scores in the face of word and sentence segmentation errors, the tool utilizes information from a word-level alignment between the yields of the system parses and reference parses in a transcript chunk (e.g., a conversation side or story)².

²The tool can also provide scores based on all of the head dependencies extracted from the system and reference trees, as well as a more focused set of open class dependencies, which involve open class content words.

We extended the functionality of this tool to support the scoring of Chinese parses when the word segmentation of the input to the parser differs from the gold standard word segmentation in order to investigate the effect of word segmentation algorithms on parse quality. For this chapter, we use bracket scores with word alignment as the metric for evaluating word segmentation impact. To score parses with different word segmentations using this tool, we align the words in a gold standard file with the words in a test file and score the brackets with punctuation removed³.

4 Parsing Model

The Berkeley parser (Petrov et al., 2006; Petrov and Klein, 2007b) is an efficient and effective parser that introduces latent annotations (Matsuzaki et al., 2005; Prescher, 2005) to refine the syntactic categories and learns PCFG grammars based on these latent annotations. We have re-implemented⁴ and enhanced the Berkeley parser to handle unknown Chinese words efficiently, parse difficult sentences robustly, and operate more efficiently. We will describe our enhancements in detail in the remainder of this section.

4.1 Unknown Word Handling

The Chinese word formation process can be quite complex (Packard, 2000), and it differs substantially from the English process. Although Chinese morphology is generally considered less informative of the part-of-speech (POS) type of the entire word, its characters do reflect some information about a Chinese word. The last characters in a Chinese word are, in some cases, most informative of the POS type, while for others, it is the characters at the beginning or the middle of a word.

The Berkeley parser has some built-in ability to handle certain classes of unknown Chinese words such as digits and dates. For words outside of these classes, the parser ignores character-level information and relies only on the rare-word part-of-speech

³When evaluating test parses that have the same word segmentation as the gold standard parses, *Sparseval* provides scores on a sentence-by-sentence basis, just like EVALB.

⁴The major motivation for re-implementation is to allow more generic and flexible state-tying operations that are important for some algorithms we are developing.

tag statistics. In our approach, which is similar to (Huang et al., 2007), we employ a rather simple but effective method to estimate the word emission probability, $p(w|s)$, of an unknown word w given the latent state s of some syntactic category. We use the geometric average of the emission probability of the characters in the word, i.e., $p(c_k|s)$ with c_k being the k -th character of the word, to estimate the word emission probability:

$$P(w|s) = \sqrt[n]{\prod_{c_k \in w, P(c_k|s) \neq 0} P(c_k|s)}$$

where

$$n = |\{c_k \in w | P(c_k|s) \neq 0\}|$$

Characters unseen in the training data are ignored in the computation of the geometric average. In case there is any character in the word that was previously seen in the training data but only with other latent annotations, then we back off by using the rare word statistics from the state s regardless of the word.

As we can see in Table 2, the character based unknown word handling method improves performance on both recall and precision when evaluated on CTB5.

Treebank	Unk	Recall	Precision	F
CTB5	default	82.12	82.88	82.50
	character	82.84	83.22	83.03

Table 2: The effect of the character-based unknown word handling method.

4.2 Robustness

Although the grammars trained by the Berkeley parser are compact, parsing can still be computationally expensive because of the many fine-grained latent states to consider in the computation of inside-outside probabilities in the chart. Fortunately, Petrov and Klein (Petrov and Klein, 2007b) developed an efficient coarse-to-fine parsing strategy that starts from simpler grammars and prunes away unlikely chart items before parsing using the more complex grammars. In practice, the pruning thresholds are chosen to balance the parsing accuracy and speed.

Setting pruning thresholds to support robust but efficient parsing can be challenging over the range of

parsing tasks we support in the GALE project; our parser is expected to parse various different materials, ranging from newswire text, which tends to be grammatically well formed, to n -best ASR outputs, many of which are poorly formed sentences. It is not uncommon for thresholds that work well for parsing treebank materials fail to parse less well-formed sentences we are expected to process. To support more robust parsing, we incrementally lower pruning thresholds when the current thresholds fail to parse a sentence and then restore the default thresholds after finishing parsing that sentence. With this strategy, we were able to more fully parse sentences in the GALE processing pipeline. We balance this with a parsing time threshold to address cases when sentences are too long to parse in a reasonable amount of time, returning an empty parse when the time threshold is exceeded.

In addition to variable pruning thresholds, we made several other minor changes to the parser to improve robustness. For example, we added smoothing for the word emission probabilities associated with a word tag pair w, t to eliminate zeros when a frequently observed word w does not occur with tag t in the training data.

4.3 Parallelization

The time needed to train a single-threaded Berkeley parser is acceptable for a corpus of moderate size. For example, it takes roughly one day to train a grammar on CTB5+CTB6BN. However, the training speed becomes a bigger issue when applying self-training strategies. For example, in the self-training experiments described in Section 8, the automatically labeled training data is an order of magnitude larger than the treebank training set and it takes several weeks to finish training a more complex grammar. In our reimplementa-tion of the parser, we parallelized the Expectation-Maximization step, the most computationally intensive component of the training code to take advantage of the computing power of multi-core/multi-CPU machines. The resulting training code is about 7 times faster with 10 threads on a 16-core machine than the single threaded version. We parallelized the parsing code as well to improve the decoding speed. The MapReduce framework (Dean and Ghemawat, 2005) is also naturally applicable to both grammar

training and sentence parsing, and this framework will be explored in future work.

5 Word Segmentation and Parsing

Written Chinese text consists of sequences of characters with no delimiters between words, and yet for word-based NLP applications (tagging, parsing, MT), word segmentation is a prerequisite. The output of word segmentation algorithms may vary depending on their different definitions of words and system engineering requirements. While word segmentation, in and of itself, is worthy of study, as shown in the efforts in the SIGHAN Chinese Word Segmentation Bakeoffs (Sproat and Emerson, 2003), in this section we focus on the effect of word segmentation on parse accuracy. Discrepancies in word segmentation compared to the gold standard are likely to be the source of significant error when comparing the parse of those words to the gold standard parse of the gold standard word segmentation. Consistency between the words in the gold standard trees and the input to the parser is likely to be an important factor for achieving greater parse accuracies.

Here we will use the Penn Chinese Treebank (CTB) Segmentation Standard (Xia, 2000) because that is the resource we use to train our Chinese parser (Xue et al., 2002; Xue et al., 2005). In the rest of this section we will first describe the word segmentation algorithms we use and then evaluate the effect of those algorithms on parse accuracy. As described in section 3, the Sparseval tool with alignment will be used to score parses because there can be mismatches between the yields of the test parses and gold standard trees.

5.1 Word Segmentation Algorithms

Parse accuracy can be substantially affected by the word segmentation algorithm and how it is trained. A word segmentation algorithm that is trained on treebank data is likely to produce a word segmentation that is more consistent with the treebank, and so is likely to result in parses with greater parse accuracy. We evaluated three different segmenters.

The first is Fair Issac’s extension of the LDC segmenter (LNUplus) with an expanded dictionary (it adds words from the NYU name dictionary and the lexicon used by the University of Washington word

segmenter), and instead of using a greedy left-to-right, longest dictionary entry match at each point, it uses dynamic programming to find a segmentation that maximizes mean token length. This is strictly a dictionary-driven segmenter, and the dictionaries it uses were not optimized to conform with CTB6.

The University of Washington word segmenter (UW) (Hwang et al., 2006) uses a very large segmentation dictionary that includes frequent words extracted from GALE acoustic data transcripts, Chinese Gigaword 2, and the Chinese treebank 6.0. It then uses the longest-first-match algorithm to segment the training data comprised of GALE acoustic data transcripts and Chinese Gigaword, and then trains an ngram LM on these sources and the Chinese treebank 6.0 data duplicated 5 times to weight the source more heavily. The language model uses a vocabulary of 70K words, with the remaining words mapped to a garbage word. This ngram LM is then used to provide the maximum likelihood segmentation for a sentence. This word segmenter uses information provided by CTB6, but it is not optimized for the treebank.

The Stanford Chinese Word Segmenter (Tseng et al., 2005; Chang et al., 2008) uses conditional random fields with character identity, morphological, and character reduplication features extracted from the training data. The current segmenter also uses external lexicon features (Chang et al., 2008) to segment more consistently. We trained several versions of the Stanford segmenter:

- **Stanford (all)** is trained using the entire Chinese Penn Treebank. This represents an upper bound on the parsing accuracy for the parser given this word segmentation approach (since the training set contains the test sentences).
- **Stanford (parser)** is trained using the same training data split of the Chinese Penn Treebank as the parser, i.e., the CTB5+CTB6BN training set. The same dev set in Table 1 is used for development.
- **Stanford (parser+LDC)** is trained under the same conditions as Stanford (parser) except that the training data is augmented with an additional file that was segmented by LDC and contains 16,448 sentences.

- **Stanford (parser+LDC+self-labeled)** is trained under the same conditions as Stanford (parser+LDC) except that the training data is augmented with Gale data files⁵ that were segmented using Stanford (parser). Hence, this segmenter was trained on 175,852 sentences in total (24,416 from CTB5+CTB6BN training, 16,448 LDC hand-segmented, and 134,988 automatically word segmented).
- **Stanford (parser+LDC+UW-labeled)** is trained under the same conditions as Stanford (parser+LDC) except that the training data is augmented with the same Gale data files as in Stanford (parser+LDCself-labeled) segmented using the UW segmenter.

5.2 Results and Discussion

In Table 3, we compare the word segmentation and parsing performance of each of the segmentation algorithms and training conditions described in Section 5.1. The parser is our reimplementation of the Berkeley parser described in Section 4. The parser was trained on the CTB5+CTB6BN training set (i.e., treebank data only), and this parser was used to parse the CTB5 test set re-segmented by each of the word segmentation models. The resulting parses were then compared with the gold standard using aligned bracket scoring from the Sparseval tool as described in Section 3.

It is not surprising that the best parsing performance is obtained using the treebank’s gold standard word segmentation. Parses of the sentences

⁵The files used for self-training are

- GALE-Kickoff-bc_LDC2005E63-refs,
- GALE-P2R1-refs-LDC2007E05_bc_quick_rich,
- GALE-P2R1-refs-LDC2007E05_bn_quick_rich,
- GALE-P2R2-refs-LDC2007E45_bc_quick_rich,
- GALE-P2R2-refs-LDC2007E45_bn_quick_rich,
- GALE-P2R3-refs-LDC2007E86_bc_quick_rich,
- GALE-P2R3-refs-LDC2007E86_bn_quick_rich,
- GALE-Y1Q1-bc_LDC2005E8-manual-trans-QRTR-refs,
- GALE-Y1Q1-bc_LDC2005E8-manual-trans-QTR-refs,
- GALE-Y1Q1-bn_LDC2005E82-manual-trans-QRTR-refs,
- GALE-Y1Q1-bn_LDC2005E82-manual-trans-QTR-refs,
- GALE-Y1Q2-bc_LDC2006E33-manual-trans-QRTR-refs,
- GALE-Y1Q2-bn_LDC2006E33-manual-trans-QRTR-refs,
- GALE-Y1Q3-refs-LDC2006E84_bc_quick_rich,
- GALE-Y1Q3-refs-LDC2006E84_bn_quick_rich,
- GALE-Y1Q4-refs-LDC2006E91_bn_quick_rich.

processed by the LNUplus and UW segmenters had significantly⁶ lower parse scores than was obtained given the gold standard word segmentation. Since neither segmenter was tuned to the treebank’s word segmentation standard, the errors in word segmentation harm the parse accuracies greatly. The UW segmenter’s use of the treebank data improves its word segmentation accuracy relative to the LNUplus segmenter; however, the use of training data that is inconsistent with the treebank’s standard plays a role in the 10% drop in parse accuracy.

To get at the upper bound parsing performance of the Stanford segmenter, we evaluated parsing accuracy when it was trained on the entire treebank. Although parse accuracy in this case is slightly lower than that obtained with gold standard word segmentation, it is clear that match to the gold standard words is an important factor for obtaining accurate Chinese parses. The Stanford segmenter was able to produce fairly accurate word segmentations when trained on the same treebank data as the parser and achieved parse F-scores within 3% of those obtained using gold standard word segmentations. It is notable that retraining the Stanford segmenter using additional LDC-segmented data and self-labeled data improves both word segmentation and parsing scores. When adding automatically labeled data to the training set, it is important that it is consistent with the treebank word segmentation standard, as can be observed by the decline in performance obtained by adding data segmented by the UW segmenter. The use of consistently self-labeled data to re-train the segmenter improves parse performance by 1% F-measure over using the treebank training data alone. If parsing accuracy is important for downstream applications, then using a word segmenter that is tuned to the treebank standard is vital for achieving performance levels that are within 1-2% of those obtained with perfect word segmentation.

6 Text Tokenization

The UW Decatur (Zhang and Kahn, 2008) text normalization process was developed to standardize the

⁶We used Bikel’s randomized parsing evaluation comparator to determine the significance ($p < 0.05$) of difference between two parsers’ output.

Models	Word Segmentation			Parser		
	Recall	Precision	F	Recall	Precision	F
Trebank Segmentation	100	100	100	83.58	84.00	83.79
LNUplus	81.70	82.50	82.10	67.46	68.29	67.87
UW	86.60	91.60	89.00	71.15	76.71	73.83
Stanford (all)	99.40	99.40	99.40	83.12	83.45	83.28
Stanford (parser)	95.70	96.60	96.20	80.49	81.65	81.07
Stanford (parser+LDC)	96.10	96.90	96.50	80.59	81.75	81.16
Stanford (parser+LDC+self-labeled)	97.50	97.60	97.50	81.84	82.27	82.05
Stanford (parser+LDC+UW-labeled)	89.00	90.50	89.80	74.13	75.98	75.04

Table 3: Results on the Chinese Treebank 6.0 test set for different models and training configurations.

text pipeline stream for the NightingALE team. The value of this normalization for parsing lies with the fact that a parser that is trained to match its input conditions will perform better than a parser that must process highly varying data that is harder to model. For example, if we train the parser on the treebank, which largely contains full-width punctuation, but the input to the parser contains only half-width punctuation, then the parsing performance would be quite poor due to the fact that the treebank contains only a few half-width punctuation tokens⁷. While it is a simple matter to create a second set of training trees with all punctuation converted to half-width and mix it with the original treebank training, it is unclear that this would cover all the ways in which full and half-width punctuation occur in real-world data. However, it is a simple matter to apply the Decatur normalization to our treebank by converting all full-width letters, digits, and punctuation marks to their half-width equivalents and train on the normalized trees. The Decatur tokenizer would then be applied to all input coming through the NightingALE pipeline to ensure that we have a good match between training and test conditions.

To ensure that the Decatur tokenization does not harm parsing performance significantly, we compare a parser trained on the original treebank CTB5 training set and tested on the original CTB5 test set with one trained based on Decatur normalized training

and tested on the Decatur normalized CTB5 test set. As can be seen in Table 4, the parsing performance is slightly degraded by the normalization process, largely due to the fact that some punctuation distinctions are collapsed. However, because Decatur normalization will be carried out on all input to the parser, it eliminates a lot of the variability in the input that it would be hard to anticipate to adequately train the parser for the Gale pipeline.

CTB5	Recall	Precision	F
Original	82.84	83.22	83.03
Decatur	82.66	83.14	82.90

Table 4: Performance of the parser on CTB5 before and after Decatur normalization.

7 Genre Mapping

Newswire stories contain a wide variety of textual phenomena, including words, symbolic expressions, and punctuation, using both full and half-width representations. When parsing newswire text, it is beneficial to normalize the input to the parser and map the training trees to the same normalized style to match the conditions in which the parser will be used. Additionally, it is important to fully utilize existing treebank resources to train the parser, given the limited amount of training data. Hence, we investigate the effect of combining trees from the broadcast news genre with the newswire training trees when training a newswire model. Table 5 shows the benefit of adding text normalized broadcast news data for training a better grammar for newswire. As we can see, the grammar trained on the combination of CTB5 and CTB6BN has a significantly better performance when evaluated on the CTB5 test set, due

⁷If we accidentally parse sentences using a model trained with a mismatching representation of punctuation, say the sentence contains full-width punctuation but the parser was trained with parses containing half-width punctuation, then the parse accuracy would be quite low, dropping from 82.90 to 72.16 F score. This poor level of parse accuracy results because the punctuation marks are not recognized as such.

to the increase in domain matched training data. In an attempt to further boost performance, we will investigate the effect of combining automatically parsed newswire genre-matched trees with treebank training trees to further improve the newswire model in Section 8.

Training Data	Recall	Precision	F
CTB5	82.66	83.14	82.90
CTB5+CTB6BN	83.36	83.95	83.65

Table 5: Results on the Chinese CTB5 test set for grammars trained on different configurations.

When parsing automatically transcribed broadcast conversation speech, the parser will need to process speech transcripts that are segmented into SUs using automatic sentence boundary detection; hence, words will be fully verbal (i.e., no symbolic expressions) and there will be no punctuation. While there is a small treebank that matches this genre, access to a greater diversity of speech-based trees is important for training as accurate a model as possible.

Because the annotated BC treebank is quite limited in size, a grammar trained only on BC treebank alone is quite poor as can be observed in Table 6. It is a natural idea to use CTB6 to enhance the training corpora for BC grammars. However, the textual nature of the CTB5 sub-corpus presents a challenge. One obvious difference is that the transcripts of the BC treebank are verbalized, i.e., all numbers, dates, and other phenomena are orthographically transcribed while CTB5 contains many symbolic forms. Although CTB6BN should be orthographically transcribed as speech, it contains digits and other symbolic expressions that make it inconsistent with the BC treebank. To address the mismatch between the CTB6 and BC treebanks in order to use CTB6 as additional training material for BC grammars, we have developed scripts to speech normalize (i.e., verbalize) the CTB6 treebank. The normalization procedure considers the part-of-speech tag of a word, as well as its context to determine how to verbalize the word. The major normalization operations include:

- Verbalize all sequences of digits based on their context. If the context suggests that a sequence

of digits represents a number, this number is verbalized according to how it is pronounced, e.g., 12.4% is verbalized to 百分之十二点四 (twelve point four percent). Otherwise each digit is verbalized as it is pronounced individually, e.g., 03012 is verbalized to 零三零一二 (zero three zero one two).

- Special care is taken to verbalize digits in temporal nouns. For example, 1998 is verbalized to 一九九八年 (year nineteen ninety eight), 5.4 to 五四 (The May 4th Movement), 10:12 to 十二分十二秒 (ten minutes twelve seconds), etc.
- Split foreign names with first and last names delimited by a dot or hyphen into two names. Web and email addresses are also split into separate tokens with ‘.’ and ‘@’ verbalized.
- Remove all punctuation.

Table 6 shows the benefit of adding the speech normalized CTB6 for training a better grammar for conversational speech. As we can see, although the grammar trained on speech normalized⁸ CTB6 has a much lower performance when evaluated on the BC test set, its combination with the small amount of in-domain BC training data boosts the performance significantly, outperforming the grammar trained on BC training data alone. We will also investigate the use of additional automatically parsed BC genre-matched sentences in the self-training manner to further improve parsing performance on BC data.

Training Data	Recall	Precision	F
BC	76.89	76.48	76.69
CTB5+CTB6BN	72.85	71.66	72.25
BC+CTB5+CTB6BN	78.95	78.96	78.96

Table 6: Results on the Chinese BC test set for grammars trained on different configurations.

8 Parser Self-Training

To support self-training in a semi-supervised setting, manually labeled training data is used to train

⁸If we had not normalized the treebank data for speech, the parsing accuracy would have been far worse due to a mismatch between the training and testing conditions.

an initial parser, which is then used to parse additional unlabeled data to combine with the manually labeled data to retrain a new parser. Early investigations using self-training for parser training were fairly unsuccessful at improving in-domain parsing. Charniak (1997) reported that self-training does not improve the performance of a context-free grammar trained on the WSJ training set. Steedman et al. (2003) reported some degradation using a lexicalized tree adjoining grammar parser (Sarkar, 2001) and a minor gain using Collins lexicalized PCFG parser (Collins, 1999); however, this gain was obtained using a poor parser trained on few sentences and was expected to level out quickly on a larger training set. One might conclude from these investigations that either the self-labeled data does not provide useful information or the training algorithm used for the parsers does not learn useful information from the self-labeled data.

Recently, McClosky et al. (2006) used semi-supervised training of a reranking parser (Charniak and Johnson, 2005) and obtained exciting positive results on WSJ over a strong baseline. Their state-of-the-art reranking parser consists of two components, a lexicalized probabilistic 50-best parser and a discriminative reranker, which re-orders and selects the best of the 50-best parses returned by the first generative parser by utilizing a rich set of features that could not be feasibly used by the first parser. They used this two-stage parser to parse millions of unlabeled news article sentences and retrained the first parser using the combination of the original labeled data and the reranker-labeled data. Although they did not fully explain why this two stage method works, they suggest that the parse trees selected by the superior reranker provides guidance for retraining the first parser, which in turn produces better 50-best parses for the reranker. It should be noted that they also reported that no improvement was obtained from self-training their generative parser.

We have investigated the self-training capability of our parser on both newswire and broadcast conversation by utilizing moderately large amount of unlabeled in-domain data. As described in Section 2, 210k unlabeled sentences are used for both the newswire and broadcast conversation genres. The results are presented in Table 7. For the NW task, self-training gains one point on F measure over

the grammar trained using the treebank CTB6 alone. This improvement is even greater than the benefit of adding CTB6BN to CTB5. The improvement on the BC task is smaller but consistent across different random runs. Given that the number of words for self-training BC is much smaller than for the NW task, use of additional genre-matched data is an important next step. All the improvements are statistically significant. We also tried self-training using a Chinese port of Charniak’s generative parser on our data set but obtained no significant improvement.

Genre	Training Data	Recall	Precision	F
NW	CTB6	83.36	83.95	83.65
	+unlabeled	84.28	85.28	84.78
BC	BCTB+CTB6	78.95	78.96	78.96
	+unlabeled	79.54	79.44	79.49

Table 7: The performance of self-training on both NW and BC.

We observed that many of the rule parameters of the grammar trained on CTB6 alone have zero probability. On one hand, this is what we want because the grammar should learn about impossible rule expansions. On the other hand, this might also be a sign of over-fitting. In contrast, the grammar obtained using self-training contains much more non-zero rules than the grammar trained on CTB6 alone. This suggests that one of the benefits of using automatically labeled data is smoothing. The nature of the EM algorithm aims to adjust the model parameters to increase the likelihood of the training data. The greater the number of free parameters, the more power EM has to learn from and fit the training data. Resulting grammars may not generalize well when the training data is too small in size. We believe that the addition of automatically labeled data helps to prevent the EM algorithm from over-fitting the correctly labeled training data while learning with more latent states. This hypothesis will be investigated further in future work.

9 Conclusions

In this chapter, we have explored issues such as unknown word handling and word segmentation in parsing Chinese, and conducted experiments that highlight the fact that greater newswire parse accuracy can be achieved by training on the combina-

tion of newswire and broadcast news parses, than by training on newswire parses alone. Additionally, we have found that self-training with a large amount of unlabeled data further improves parsing performance. We conjecture based on our analyses that the EM training algorithm is able to exploit the information available in both gold and automatically labeled data with more complex grammars while being less affected by over-fitting the treebank. Self-training should also benefit other discriminatively trained parsers with latent annotations (Petrov and Klein, 2007a; Petrov and Klein, 2008), although training would be much slower compared to using generative models, as in our case.

In future work, we will evaluate the impact of using larger quantities of self-labeled data. As the amount of data increases, it will also be important to investigate the impact of weighting the self-labeled data. It is quite possible that the errors in the automatically labeled data could limit the accuracy of the self-trained model, especially when there is a much greater quantity of automatically labeled data than the gold standard training data, so by weighting the posterior probabilities computed for the gold and automatically labeled data the information provided by the gold trees will not be swamped by the errors in the automatically labeled trees. Another approach would be to introduce the automatically labeled data at different stages of training rather than at the outset, because it is possible that a later introduction of the automatically labeled data, after well-founded annotations are learned from the treebank, would result in more effective learning. We will also investigate methods for automatic data selection of the automatically labeled sentences in order to choose those that would be most helpful for self-training.

Acknowledgments

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023 and NSF IIS-0703859. Any opinions, findings and/or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies or the institutions where the work was completed.

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*.
- Pi-Chuan Chang, Michel Gally, and Christopher Manning. 2008. Optimizing Chinese Word Segmentation for Machine Translation Performance. In *ACL 2008 Third Workshop on Statistical Machine Translation*.
- Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *ACL*.
- Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *ICAI*.
- Michael John Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.
- Jeffrey Dean and Sanjay Ghemawat. 2005. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*.
- Mary P. Harper, Bonnie J. Dorr, John Hale, Brian Roark, Izhak Shafran, Matthew Lease, Yang Liu, Matthew Snover, Lisa Yung, Anna Krasnyanskaya, and Robin Stewart. 2005. 2005 Johns Hopkins Summer Workshop Final Report on Parsing and Spoken Structural Event Detection. Technical report, Johns Hopkins Summer Workshop.
- Zhongqiang Huang, Mary Harper, and Wen Wang. 2007. Mandarin Part-of-Speech Tagging and Discriminative Reranking. *EMNLP*.
- Mei-Yuh Hwang, Xin Lei, Wen Wang, and Takahiro Shinzaki. 2006. Investigation on Mandarin Broadcast News Speech Recognition. In *ICSLP*.
- Roger Levy and Galen Andrew. 2006. Tregex and Tsurgeon: Tools for querying and manipulating tree data structures. In *LREC*.
- Roger Levy and Christopher Manning. 2003. Is it harder to parse Chinese, or the Chinese Treebank. In *ACL*.
- Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL*.
- David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *HLT-NAACL*.
- Jerome Packard. 2000. *The Morphology of Chinese*. Cambridge University Press.
- Slav Petrov and Dan Klein. 2007a. Discriminative Log-Linear Grammars with Latent Variables. In *NIPS*.
- Slav Petrov and Dan Klein. 2007b. Improved Inference for Unlexicalized Parsing. In *HLT-NAACL*.

- Slav Petrov and Dan Klein. 2008. Sparse Multi-Scale Grammars for Discriminative Latent Variable Parsing. In *EMNLP*.
- Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL*.
- Detlef Prescher. 2005. Inducing Head-Driven PCFGs with Latent Heads: Refining a Tree-Bank Grammar for Parsing. *ECML*.
- Brian Roark, Mary Harper, Yang Liu, Robin Stewart, Matthew Lease, Matthew Snover, Izhak Shafran, Bonnie J. Dorr, John Hale, Anna Krasnyanskaya, and Lisa Yung. 2006. SParseval: Evaluation Metrics for Parsing Speech. In *LREC*.
- Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *NAACL*.
- Richard Sproat and Tom Emerson. 2003. The first international Chinese word segmentation bakeoff. In *Proceedings of SIGHAN 2*.
- Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *EACL*.
- Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A Conditional Random Field Word Segmenter. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*.
- Fei Xia, 2000. *The Segmentation Guidelines for the Penn Chinese Treebank (3.0)*.
- Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated Chinese corpus. In *ACL*.
- Nianwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*.
- Bin Zhang and Jeremy G. Kahn. 2008. Evaluation of Decatur Text Normalizer for Language Model Training. Technical report, University of Washington.