# Heterogeneous Parsing via Collaborative Decoding

**Muhua Zhu** **Jingbo Zhu** **Tong Xiao**
Natural Language Processing Lab.
Northeastern University
zhumuhua@gmail.com
{zhujingbo, xiaotong}@mail.neu.edu.cn

## Abstract

There often exist multiple corpora for the same natural language processing (NLP) tasks. However, such corpora are generally used independently due to distinctions in annotation standards. For the purpose of full use of readily available human annotations, it is significant to simultaneously utilize multiple corpora of different annotation standards. In this paper, we focus on the challenge of constituent syntactic parsing with treebanks of different annotations and propose a collaborative decoding (or co-decoding) approach to improve parsing accuracy by leveraging bracket structure consensus between multiple parsing decoders trained on individual treebanks. Experimental results show the effectiveness of the proposed approach, which outperforms state-of-the-art baselines, especially on long sentences.

## 1 Introduction

Recent years have seen extensive applications of machine learning methods to natural language processing problems. Typically, increase in the scale of training data boosts the performance of machine learning methods, which in turn enhances the quality of learning-based NLP systems (Banko and Brill, 2001). However, annotating data by human is expensive in time and labor. For this reason, human-annotated corpora are considered as the most valuable resource for NLP.

In practice, there often exist more than one corpus for the same NLP tasks. For example, for constituent syntactic parsing (Collins, 1999; Charniak, 2000; Petrov et al., 2006) in Chinese, in addition to the most popular treebank Chinese Treebank (CTB) (Xue et al., 2002), there are also other treebanks such as Tsinghua Chinese Treebank (TCT) (Zhou, 1996). For the purpose of full use of readily available human annotations for the same tasks, it is significant if such corpora can be used jointly. At first sight, a direct combination of multiple corpora is a way to this end. However, corpora created for the same NLP tasks are generally built by different organizations. Thus such corpora often follow different annotation standards and/or even different linguistic theories. We take CTB and TCT as a case study. Although both CTB and TCT are Chomskian-style treebanks, they have annotation divergences in at least two dimensions: a) CTB and TCT have dramatically different tag sets, including parts-of-speech and grammar labels, and the tags cannot be mapped one to one; b) CTB and TCT have distinct hierarchical structures. For example, the words "中国 (Chinese) 传统 (traditional) 文化 (culture)" are grouped as a flat noun phrase according to the CTB standard (right side in Fig. 1), but in TCT, the last two words are instead grouped together beforehand (left side in Fig. 1). The differences cause such treebanks of different annotations to be generally used independently. This paper is dedicated to solving the problem of how to use jointly multiple disparate treebanks for constituent syntactic parsing. Hereafter, treebanks of different annotations are



*Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1344–1352,
Beijing, August 2010

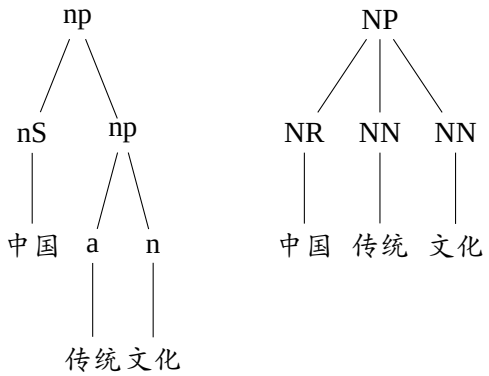
called *heterogeneous treebanks*, and correspondingly, the problem of syntactic parsing with heterogeneous treebanks is referred to as *heterogeneous parsing*.

Previous work on heterogeneous parsing is often based on treebank transformation (or treebank conversion) (Wang et al., 1994; Niu et al., 2009). The basic idea is to transform annotations of one treebank (source treebank) to fit the standard of another treebank (target treebank). Due to divergences of treebank annotations, such transformation is generally achieved in an indirect way by selecting transformation results from the output of a parser trained on the target treebank. A common property of all the work mentioned above is that transformation accuracy is heavily dependent on the performance of parsers trained on the target treebank. Sometimes transformation accuracy is not so satisfactory that techniques like instance pruning are needed in order to refine transformation results (Niu et al., 2009).

We claim there exists another way, interesting but less studied for heterogeneous parsing. The basic idea is that, although there are annotation divergences between heterogenous treebanks, actually we can also find consensus in annotations of bracket structures. Thus we would like to train parsers on individual heterogeneous treebanks and guide the parsers to gain output with consensus in bracket structures as much as possible when they are parsing the same sentences.

To realize this idea, we propose a generic collaborative decoding (or co-decoding) framework where decoders trained on heterogeneous treebanks can exchange consensus information between each other during the decoding phase. Theoretically the framework is able to incorporate a large number of treebanks and various functions that formalize consensus statistics.

Our contributions can be summarized: 1) we propose a co-decoding approach to directly utilizing heterogeneous treebanks; 2) we propose a novel function to measure parsing consensus between multiple decoders. We also conduct experiments on two Chinese treebanks: CTB and TCT. The results show that our approach achieves promising improvements over baseline systems which make no use of consensus information.




中国 传统 文化
(Chinese) (traditional) (culture)

Figure 1: Example tree fragments with TCT (left) and CTB (right) annotations

## 2 Collaborative Decoding-based Heterogeneous Parsing

### 2.1 Motivation

This section describes the motivation to use co-decoding for heterogeneous parsing. We first use the example in Fig. 1 to illustrate what consensus information exists between heterogenous treebanks and why such information might help to improve parsing accuracy. This figure contains two partial parse trees corresponding to the words "中国 (Chinese) 传统 (traditional) 文化 (culture)", annotated according to the TCT (left side) and CTB (right side) standards respectively. Despite the distinctions in tag sets and bracket structures, these parse trees actually have partial agreements in bracket structures. That is, not all bracket structures in the parse trees are different. Specifically put, although the internal structures of the parse trees are different, both CTB and TCT agree to take "中国 传统 文化" as a noun phrase. Motivated by this observation, we would like to guide parsers that are trained on CTB and TCT respectively to verify their output interactively by using consensus information implicitly contained in these treebanks. Better performance is expected when such information is considered.

A feasible framework to make use of consensus information is n-best combination (Henderson and Brill, 1999; Sagae and Lavie, 2006; Zhang et al., 2009; Fossum and Knight, 2009). In contrast

to previous work on n-best combination where multiple parsers, say, Collins parser (Collins, 1999) and Berkeley parser (Petrov et al., 2006) are trained on the same training data, n-best combination for heterogeneous parsing is instead allowed to use either a single parser or multiple parsers which are trained on heterogeneous treebanks. Consensus information can be incorporated during the combination of the output (n-best list of full parse trees following distinct annotation standards) of individual parsers. However, despite the success of n-best combination methods, they suffer from the limited scope of n-best list. Taking this into account, we prefer to apply the co-decoding approach such that consensus information is expected to affect the entire procedure of searching hypothesis space.

### 2.2 System Overview

The idea of co-decoding is recently extensively studied in the literature of SMT (Li et al., 2009; Liu et al., 2009). As the name shows, co-decoding requires multiple decoders be combined and proceed collaboratively. As with n-best combination, there are at least two ways to build multiple decoders: we can either use multiple parsers trained on the same training data (use of diversity of models), or use a single parser on different training data (use of diversity of datasets) [1]. Both ways can build multiple decoders which are to be integrated into co-decoding. For the latter case, one method to get diverse training data is to use different portions of the same training set. In this study we extend the case to an extreme situation where heterogeneous treebanks are used to build multiple decoders.

Fig. 2 represents a basic flow chart of heterogeneous parsing via co-decoding. Note that here we discuss the case of co-decoding with only two decoders, but the framework is generic enough to integrate more than two decoders. For convenience of reference, we call a decoder without incorporating consensus information as *baseline decoder*

[1]To make terminologies clear, we use *parser* as its regular sense, including training models (ex. Collins model 2) and parsing algorithms (ex. the CKY algorithm used in Collins parser), and we use *decoder* to represent parsing algorithms with specified parameter values

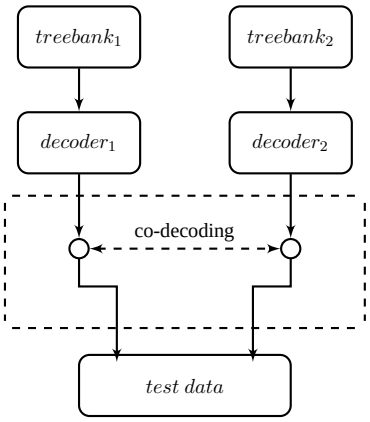



Figure 2: Basic flow chart of co-decoding

and correspondingly refer to a decoder augmented with consensus information as *member decoder*. So the basic steps of co-decoding for heterogeneous parsing is to first build baseline decoders on heterogeneous treebanks and then use the baseline decoders to parse sentences with consensus information exchanged between each other.

To complete co-decoding for heterogeneous parsing, three key components should be considered in the system:

- Co-decoding model. A co-decoder consists of multiple member decoders which are baseline decoders augmented with consensus information. Co-decoding model defines how baseline decoders and consensus information are correlated to get member decoders.

- Decoder coordination. Decoders in the co-decoding model cannot proceed independently but should have interactions between each other in order to exchange consensus information. A decoder coordination strategy decides on when, where, and how the interactions happen.

- Consensus-based score function. Consensus-based score functions formalize consensus information between member decoders. Taking time complexity into consideration, consensus statistics should be able to be computed efficiently.

In the following subsections, we first present the generic co-decoding model and then describe in detail how member decoders collaborate. Finally we introduce a novel consensus-based score function which is used to quantify consensus information exchanged between member decoders.

### 2.3 Generic Co-decoding Model

The generic co-decoding model described here is also used in (Li et al., 2009) for co-decoding of machine translators. For a given sentence $S$, a parsing algorithm (decoder) seeks a parse tree $T^*$ which is optimal in the sense that it maximizes some score function $F(T)$, as shown in Eq. 1.

$$T^* = \underset{Ts.t.S=yield(T)}{\arg\max} F(T) \quad (1)$$

where $Ts.t.S = yield(T)$ represents the set of parse trees that yield the input sentence $S$. For baseline decoders, the score function $F(T)$ is generally just the inside probability $P(T)$ [2] of a tree $T$, defined as the product of probabilities of grammar rules appearing in parse tree $T$: $\prod_{r\in R(T)} P(r)$. In the co-decoding framework, $F(T)$ is extended so as to integrate consensus-based score functions which measure consensus information between member decoders, as shown in Eq. 2.

$$F_m(T) = P_m(T) + \sum_{k,k\neq m}^{n} \Psi_k(H_k(S), T) \quad (2)$$

We use $d_k$ to denote the $k_{th}$ decoder and use $H_k(S)$ to denote corresponding parsing hypothesis space of decoder $d_k$. Moreover, $P_m(T)$ is referred to as *baseline score* given by baseline decoders and $\Psi_k(H_k(S), T)$ is *consensus score* between decoders $d_m$ and $d_k$, which is defined as a linear combination of consensus-based score functions, as shown in Eq. 3.

$$\Psi_k(H_k(S), T) = \sum_l \lambda_{k,l} f_{k,l}(H_k(S), T) \quad (3)$$

where $f_{k,l}(H_k(S), T)$ represents a consensus-based score function between $T$ and $H_k(S)$, and $\lambda_{k,l}$ is the corresponding weight. Index $l$ ranges over all consensus-based score functions in Eq. 3. Theoretically we can define a variety of consensus-based score functions.

For the simplest case where there are only two member decoders and one consensus-based score function, Eq. 2 and Eq. 3 can be combined and simplified into the equation

$$F_i(T) = P_i(T) + \lambda_{1-i} f(H_{1-i}(S), T) \quad (4)$$

where index $i$ is set to the value of either 1 or 0. This simplified version is used in the experiments of this study.

### 2.4 Decoder Coordination

This subsection discusses the problem of decoder coordination. Note that although Eq. 2 is defined at sentence level, the co-decoding model actually should be applied to the parsing procedure of any subsequence (word span) of sentence $S$. So it is natural to render member decoders collaborate when they are processing the same word spans. To this end, we would like to adopt best-first CKY-style parsing algorithms as baseline decoders, since CKY-style decoders have the property that they process word spans in the ascending order of span sizes. Moreover, the hypotheses [3] spanning the same range of words are readily stacked together in a chart cell before CKY-style decoders move on to process other spans. Thus, member decoders can process the same word spans collaboratively from small ones to big ones until they finally complete parsing the entire sentence.

A second issue in Eq. 2 is that consensus-based score functions are dependent on hypothesis space $H_k(S)$. Unfortunately, the whole hypothesis space is not available most of the time. To address this issue, one practical method is to approximate $H_k(S)$ with a n-best hypothesis list. For best-first CKY parsing, we actually retain all unpruned partial hypotheses over the same span as the approximation. Hereafter, the approximation is denoted as $\hat{H}_k(S)$

Finally, we notice in Eq. 2 that consensus score

[2] Actually, the joint probability P(S,T) of sentence $S$ and parse tree $T$ is used, but we can prove that $P(S,T) = P(T)$.

[3] In the literature of syntactic parsing, especially in chart parsing, hypotheses is often called *edges*. This paper will continue to use the terminology *hypothesis* when no ambiguity exists.

$\Psi_k(H_k(S), T)$ and $H_k(S)$ form a circular dependency: searching for $H_k(S)$ requires both baseline score and consensus score; on the other hand, calculating consensus score needs $H_k(S)$ (its approximation in practice) to be known beforehand. Li et al. (2009) solves this dilemma with a bootstrapping method. It starts with seedy n-best lists generated by baseline decoders and then alternates between calculating consensus scores and updating n-best hypothesis lists. Such bootstrapping method is a natural choice to break down the circular dependency, but multi-pass re-decoding might dramatically reduce decoding efficiency. Actually, Li et al. (2009) restricts the iteration number to two in their experiments. In this paper, we instead use an alternative to the bootstrapping method. The process is described as follows.

1. In traditional best-first CKY-style parsing algorithms, hypotheses over the same word spans are grouped according to some criterion of hypothesis equivalence [4]. Among equivalent hypotheses, only a single optimal hypothesis is retained. In this paper, we instead keep top $k$ of equivalent hypotheses in a data structure called *best-first cache*.

2. Use hypotheses in best-first caches to approximate $H_k(S)$, and calculate consensus score $\Psi_k(H_k(S), T)$ between decoders.

3. Use baseline score and consensus score to locally rerank hypotheses in best-first caches. Then remove hypotheses in caches except the top one hypothesis.

In this study, we choose the best-first CKY-style parsing algorithm used in Collins parser (Collins, 1999). Algorithm 1 extends this algorithm for co-decoding. The first two steps initialize baseline decoders and assign appropriate POS tags to sentence $S^t$. Since baseline decoders are built on heterogeneous treebanks, POS taggers corresponding to each baseline decoder are demanded, unless gold POS tags are provided. The third step is the core of the co-decoding algorithm. Here the *complete* procedure invokes baseline decoders to complete parsing on the span $[start, end]$ and generates $\hat{H}_k(s)$. The *co-decoding* procedure calculates consensus score and locally reranks hypotheses in best-first caches.

[4] the simplest criterion of equivalence is whether hypotheses have the same grammar labels.

**Algorithm 1** CKY-style Co-decoding
**Argument:** $d_k$ {the set of baseline decoders}
$S^t$ {a sentence to be parsed}

**Begin**
**Steps:**
**1.** assign POS tags to sentence $S^t$
**2.** initialize baseline decoders $d_k$
**3. for** span from 2 to sentence_length **do**
  **for** start from 1 to (sentence_length-span+1) **do**
    end := (start + span - 1)
    **for** each base decoder $d_k$ **do**
      complete($d_k$, start, end)
    **do** co-decoding(start, end)
**End**

**Subroutine:**
**complete**($d_k$, start, end): base decoder $d_k$ generates hypotheses over the span (begin.end), and fills in best-first caches.
**co-decoding**(start, end): calculate consensus score and rerank hypotheses in best-first caches. The top 1 is chosen to be the best-first hypothesis.

### 2.5 Consensus-based Score Function

There are at least two feasible ways to measure consensus between constituency parse trees. By viewing parse trees from diverse perspectives, we can either use functions on bracket structures of parse trees, as in (Wang et al., 1994), or use functions on head-dependent relations by first transforming constituency trees into dependency trees, as in (Niu et al., 2009). Although the co-decoding model is generic enough to integrate various consensus-based score functions in a uniform way, this paper only uses a bracket structure-based function.

As mentioned above, the function proposed in (Wang et al., 1994) is based on bracket structures. Unfortunately, that function is not applicable in the situation of this paper. The reason is that, the function in (Wang et al., 1994) is defined to work on two parse trees, but this paper instead needs a function on a tree $T$ and a set of trees (the approximation $\hat{H}_k(S)$). To this end, we first introduce the concept of *constituent set (CS)* of a parse tree. Conceptually, CS of a parse tree is a set of word spans corresponding to all the sub-

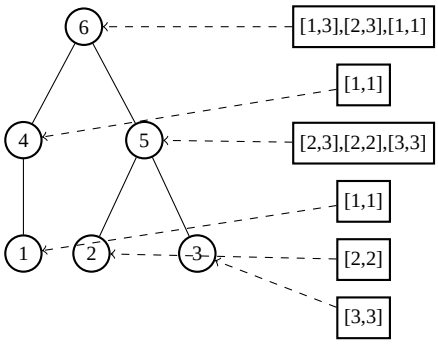



Figure 3: Constituent set of a synthetic parse tree

trees of the tree, as illustrated in Fig. 3. For example, the constituent set of the tree rooted at node 6 has three elements: $[1,1]$, $[1,3]$, and $[1,2]$. For $\hat{H}_k(S)$, the constituent set is defined as the union of constituent sets of all elements it contains.

$$CS(\hat{H}_k(S)) = \bigcup_{T \in \hat{H}_k(S)} CS(T)$$

In practice, we need to cut off elements in $CS(\hat{H}_k(S))$ in order to retain most confident word spans.

With the concept of constituent set, a consensus-based score function on $T$ and $\hat{H}_k(S)$ can be defined as follows.

$$f(\hat{H}_k(S), T) = \frac{\sum_{c \in CS(T)} I(c, CS(\hat{H}_k(S)))}{|CS(T)|} \quad (5)$$

where $I(c, CS(\hat{H}_k(S)))$ is an indicator function which returns one if $c \in CS(T)$ is compatible with all the elements in $CS(\hat{H}_k(S))$, zero otherwise. Two spans, $[a,b]$ and $[i,j]$ are said to be compatible if they satisfy one of the following conditions: 1) $i > b$; 2) $a > j$; 3) $a \leq i \leq b$ and $j \leq b$; 4) $i \leq a \leq j$ and $b \leq j$. Fig 4 uses two example to illustrate the concept of compatibility.

## 3 Experiments

### 3.1 Data and Performance Metric

The most recent version of the CTB corpus, CTB 6.0 and the CIPS ParsEval data are used as heterogeneous treebanks in the experiments. Following the split utilized in (Huang et al., 2007), we divided the dataset into blocks of 10 files. For each

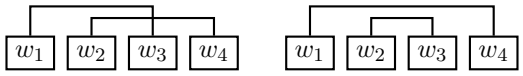



Figure 4: left) two spans conflict; right) two spans are compatible

block, the first file was added to the CTB development data, the second file was added to the CTB testing data, and the remaining 8 files were added to the CTB training data. For the sake of parsing efficiency, we randomly sampled 1,000 sentences of no more than 40 words from the CTB test set.

| CTB-Partitions | Train | Dev | Test |
|---|---|---|---|
| #Sentences | 22,724 | 2,855 | 1,000 |
| #Words | 627,833 | 78,653 | 25,100 |
| Ave-Length | 30.1 | 30.0 | 20.3 |
| TCT-Partitions | Train | Dev | Test |
| #Sentences | 32,771 | N/A | 1,000 |
| #Words | 354,767 | N/A | 10,400 |
| Ave-Length | 10.6 | N/A | 10.4 |

Table 1: Basic statistics on the CTB and TCT data

CIPS-ParsEval data is publicly available for the first Chinese syntactic parsing competition, CIPS-ParsEval 2009. Compared to CTB, sentences in CIPS-ParsEval data are much shorter in length. We removed sentences which have words less than three. CIPS-ParsEval test set has 7,995 sentences after sentence pruning. As with the CTB test set, we randomly sampled 1,000 sentences for evaluating co-decoding performance. Since CIPS-ParsEval data is actually a portion of the TCT corpus, for convenience of reference, we will refer to CIPS-ParsEval data as TCT in the following sections. Table 1 contains statistics on CTB and TCT.

The two training sets are used individually to build baseline decoders. With regard to the test sets, each sentence in the test sets should have two kinds of POS tags, according to the CTB and TCT standards respectively. To this end, we applied a HMM-based method for POS annotation transformation (Zhu and Zhu, 2009). During the POS transformation, the divergences of word segmentation are omitted.

For all experiments, *bracketing F1* is used as the performance metric, provided by *EVALB* [5].

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

### 3.2 Baseline Decoders

As already mentioned above, we apply Collins parser in this paper. Specifically speaking, two CKY-style baseline decoders to participate co-decoding are built on CTB and TCT respectively with Collins model two. For the CTB-based decoder, we use the CTB training data with slight modifications: we replaced POS tags of punctuations with specific punctuation symbols.

To get the TCT-based decoder, we made following modifications. Firstly, TCT is available with manually annotated head indices for all the constituents in parse trees. For example, a grammar label, say, np-1, means that the constituent is a noun phrase with the second child being its head child. In order to relax context independence assumptions made in PCFG, we appended head indices to grammar labels to get new labels, for example $np1$. Secondly, since Collins parser is a lexicalized parser, head rules specific to the TCT corpus were manually created, which are used together with readily available head indices. Such adaptation is also used in (Chen et al., 2009);

### 3.3 Parsing Results

We conduct experiments on both CTB and TCT test sets. Two parameters need to be set: the cut-off threshold for constructing constituent set of $\hat{H}_k(S)$ and the weight $\lambda$ [6] of consensus score in Eq. 4. We tuned the parameters on the CTB development set and finally set them to 5 and 20 respectively in the experiments. Table 2 presents bracketing F1 scores of baseline systems and the co-decoding approach. Here, the row of *baseline* represents the performance of individual baseline decoders, and the comparison of baseline and co-decoding on a test set, say CTB, demonstrates how much boosting the other side, say TCT, can supply. For the co-decoding approach, the size of best-first cache is set to 5 which achieves the best result among the cache sizes we have experimented.

As the results show, co-decoding achieves promising improvements over baseline systems on both test sets. Interestingly, we see that the improvement on the TCT test set is larger than that on the CTB test set. In general, a relatively strong decoder can improve co-decoding performance more than a relatively weak decoder does. At the first sight, the TCT-based decoder seems to have better performance than the CTB-based decoder. But if taking sentence length into consideration, we can find that the TCT-based decoder is actually relatively weak. Table 3 shows the performance of the CTB-based decoder on short sentences.

[6] We use the same $\lambda$ for both member decoders.

| Test Set | CTB | TCT |
|---|---|---|
| Baseline | 79.82 | 81.02 |
| Co-decoding | 80.33 | 81.77 |

Table 2: Baseline and Co-decoding on the CTB and TCT test sets

### 3.4 Analysis

Fig. 5 shows the bracketing F1 on the CTB test set at different settings of the best-first cache size $C$. F1 scores reach the peak before $C$ increases to 6. As a result, we set $C$ to 5 in all our experiments.

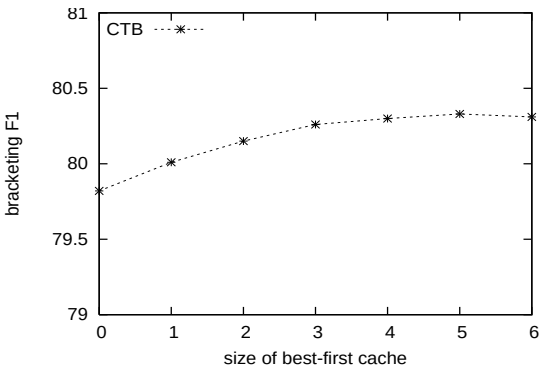



Figure 5: Bracketing F1 with varying best-first cache size

To evaluate the effect of sentence length on co-decoding, Table 3 presents F1 scores on portions of the CTB test set, partitioned according to sentence length. From the results we can see that co-decoding performs better on long sentences. One possible reason is that member decoders have more consensus on big spans. Taking this observation into consideration, one enhancement to the co-decoding approach is to enable co-decoding only on long sentences. This way, parsing ef-

| Partitions | [0,10] | (10,20] | (20,30] | (30,40] |
|---|---|---|---|---|
| # Sentence | 276 | 254 | 266 | 204 |
| Ave-Length | 6.07 | 15.64 | 25.43 | 35.20 |
| Baseline | 92.83 | 84.34 | 78.98 | 76.69 |
| Co-decoding | 92.84 | 84.36 | 79.43 | 77.65 |

Table 3: Effect of sentence length on co-decoding performance

ficiency of co-decoding can be improved. It is worth emphasizing that co-decoding is still helpful for parsers whose performance on short sentences is not satisfactory, as shown in Table 2.

Another interesting analysis is to check how many parsing results are affected by co-decoding, compared to baseline decoders. Table 4 shows the statistics.

| Test Set | # All | # Improved | # Decreased |
|---|---|---|---|
| CTB | 1000 | 225 | 109 |
| TCT | 1000 | 263 | 92 |

Table 4: Statistics on sentences of test data

As the table shows, although overall accuracy is increased, we find that on some sentences, co-decoding instead worsens parsing accuracy. In order to get insights on error sources, we manually analyzed 20 sentences on which co-decoding achieves negative results. We find a large portion (14 of 20) of sentences are short sentences (of words less than 20). Actually, due to high accuracy of the CTB-based decoder on short sentences, co-decoding is indifferent when this decoder is processing short sentences. And we also find that some errors are derived from differences in annotation standards. Fortunately, the divergence of annotations mainly exists in relatively small spans. So one solution to the problem is to enable co-decoding on relatively big spans. These will be done in our future work.

## 4 Related Work

### 4.1 System Combination

In the literature of syntactic parsing, n-best combination methods include parse selection, constituent recombination, production recombination, and n-best reranking. Henderson and Brill (1999) performs parse selection by maximizing the expected precision of selected parse with respect to the set of parses to be combined. Sagae and Lavie (2006) proposes to recombine constituents from the output of individual parsers. More recently, Fossum and Knight (2009) studies a combination method at production level. Zhang et al. (2009) reranks n-best list of one parser with scores derived from another parser.

Compared to n-best combination, co-decoding (Li et al., 2009; Liu et al., 2009) combines systems during decoding phase. Theoretically, system combination during decoding phase helps decoders to select better approximation to hypothesis space, since pruning is practically unavoidable. To the best of our knowledge, co-decoding methods have not been applied to syntactic parsing.

### 4.2 Treebank Transformation

The focus of this study is heterogeneous parsing. Previous work on this challenge is generally based on treebank transformation. Wang et al. (1994) describes a method for transformation between constituency treebanks. The basic idea is to train a parser on a target treebank and generate a n-best list for each sentence in source treebank(s). Then, a matching metric which is a function on the number of the same word spans between two trees is defined to select a best parse from each n-best list. Niu et al. (2009) applies a closely similar framework as with (Wang et al., 1994) to transform a dependency treebank to a constituency one.

## 5 Conclusions

This paper proposed a co-decoding approach to the challenge of heterogeneous parsing. Compared to previous work on this challenge, co-decoding is able to directly utilize heterogeneous treebanks by incorporating consensus information between partial output of individual parsers during the decoding phase. Experiments demonstrate the effectiveness of the co-decoding approach, especially the effectiveness on long sentences.

## Acknowledgments

This work was supported in part by the National Science Foundation of China (60873091). We would like to thank our anonymous reviewers for their comments.

## References


Banko, Michele and Eric Brill. 2001. *Scaling to very very large corpora for natural language disambiguation*. In Proc. of ACL 2001, pages 26-33.

Chen, Xiao, Changning Huang, Mu Li, and Chunyu Kit. 2009. *Better Parser Combination*. Technique Report of CIPS-ParsEval 2009.

Collins, Michael. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

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

Fossum, Victoria and Kevin Knight. 2009. *Combining constituent parsers*. In Proc. of NAACL 2009, pages 253-256.

Henderson, John and Eric Brill. 1999. *Exploiting diversity in natural language processing*. In Proc. of SIGDAT-EMNLP 1999, pages 187-194.

Huang, Zhongqiang, Mary P. Harper, and Wen Wang. 2007. *Mandarin part-of-speech tagging and discriminative reranking*. In Proc. of EMNLP-CoNLL 2007, pages 1093-1102.

Li, Mu, Nan Duan, Dongdong Zhang, Chi-Ho Li, and Ming Zhou. 2009. *Collaborative decoding: partial hypothesis re-ranking using trnaslationconsensus between decoders*. In Proc. of ACL 2009, pages 585-592.

Liu, Yang, Haitao Mi, Yang Feng, and Qun Liu. 2009. *Joint Decoding with Multiple Translation Models*. In Proc. of ACL 2009, pages 576-584.

Niu, Zheng-Yu, Haifeng Wang, Hua Wu. 2009. *Exploiting heterogeneous treebanks for parsing*. In Proc. of ACL 2009, pages 46-54.

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

Sage, Kenji and Alon Lavie. 2006. *Parser combination by reparsing*. In Proc. of NAACL 2006, pages 129-132.

Xue, Nianwen, Fu dong Chiou, and Martha Palmer. 2002. *Building a large-scale Annotated Chinese corpus*. In Proc. of COLING 2002, pages 1-8.

Wang, Jong-Nae, Jing-Shin Chang, and Keh-Yih Su. 1994. *An automatic treebank conversion algorithm for corpus sharing*. In Proc. of ACL 1994, pages 248-254.

Zhang, Hui, Min Zhang, Chew Lim Tan, and Haizhou Li. 2009. *K-best combination of syntactic parsers*. In Proc. of EMNLP 2009, pages 1552-1560.

Zhou, Qiang. 1996. *Phrase bracketing and annotating on Chinese language corpus. (in Chinese)* Ph.D. thesis, Beijing University.

Zhu, Muhua and Jingbo Zhu. 2009. *Label Correspondence Learning for Part-of-Speech Annotation Transformation*. In Proc. of CIKM 2009, pages 1461-1464.