Grammatical Relations in Chinese: GB-Ground Extraction and Data-Driven Parsing

Weiwei Sun, Yantao Du, Xin Kou, Shuoyang Ding, Xiaojun Wan[*]
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{ws,duyantao,kouxin,wanxiaojun}@pku.edu.cn, dsy100@gmail.com

Abstract

This paper is concerned with building linguistic resources and statistical parsers for deep grammatical relation (GR) analysis of Chinese texts. A set of linguistic rules is defined to explore implicit phrase structural information and thus build high-quality GR annotations that are represented as general directed dependency graphs. The reliability of this linguistically-motivated GR extraction procedure is highlighted by manual evaluation. Based on the converted corpus, we study transition-based, data-driven models for GR parsing. We present a novel transition system which suits GR graphs better than existing systems. The key idea is to introduce a new type of transition that reorders top k elements in the memory module. Evaluation gauges how successful GR parsing for Chinese can be by applying data-driven models.

1 Introduction

Grammatical relations (GRs) represent functional relationships between language units in a sentence. They are exemplified in traditional grammars by the notions of subject, direct/indirect object, etc. GRs have assumed an important role in linguistic theorizing, within a variety of approaches ranging from generative grammar to functional theories. For example, several computational grammar formalisms, such as Lexical Function Grammar (LFG; Bresnan and Kaplan, 1982; Dalrymple, 2001) and Head-driven Phrase Structure Grammar (HPSG; Pollard and Sag, 1994) encode grammatical functions directly. In particular, GRs can be viewed as the dependency backbone of an LFG analysis that provide general linguistic insights, and have great potential advantages for NLP applications, (Kaplan et al., 2004; Briscoe and Carroll, 2006; Clark and Curran, 2007a; Miyao et al., 2007).

In this paper, we address the question of analyzing Chinese sentences with deep GRs. To acquire high-quality GR corpus, we propose a linguistically-motivated algorithm to translate a Government and Binding (GB; Chomsky, 1981; Carnie, 2007) grounded phrase structure treebank, i.e. Chinese Treebank (CTB; Xue et al., 2005) to a deep dependency bank where GRs are explicitly represented. Different from popular *shallow* dependency parsing that focus on tree-shaped structures, our GR annotations are represented as general directed graphs that express not only local but also various long-distance dependencies, such as coordinations, control/raising constructions, topicalization, relative clauses and many other complicated linguistic phenomena that goes beyond shallow syntax (see Fig. 1 for example.). Manual evaluation highlights the reliability of our linguistically-motivated GR extraction algorithm: The overall dependency-based precision and recall are 99.17 and 98.87. The automatically-converted corpus would be of use for a wide variety of NLP tasks.

Recent years have seen the introduction of a number of treebank-guided statistical parsers capable of generating considerably accurate parses for Chinese. With the high-quality GR resource at hand, we study data-driven GR parsing. Previous work on dependency parsing mainly focused on structures that can be represented in terms of directed trees. We notice two exceptions. Sagae and Tsujii (2008) and Titov et al. (2009) individually studied two transition systems that can generate more general graphs rather than trees. Inspired by their work, we study transition-based models for building deep dependency structures. The existence of a large number of crossing arcs in GR graphs makes left-to-right, incremental graph spanning computationally hard. Applied to our data, the two existing systems cover only 51.0% and 76.5% GR graphs respectively. To better suit

[*]Email correspondence.

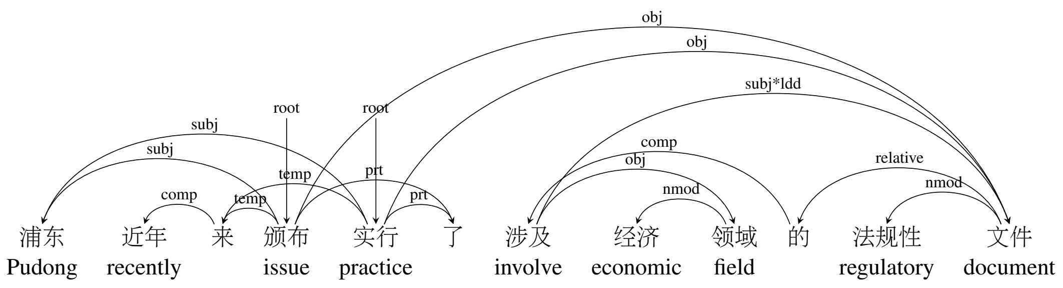

Figure 1: An example: *Pudong recently enacted regulatory documents involving the economic field*. The symbol "*ldd" indicates long-distance dependencies; "subj*ldd" between the word "涉及/involve" and the word "文件/documents" represents a long-range subject-predicate relation. The arguments and adjuncts of the coordinated verbs, namely "颁布/issue" and "实行/practice," are separately yet distributively linked the two heads.

our problem, we extend Titov et al.'s work and study what we call K-permutation transition system. The key idea is to introduce a new type of transition that reorders top k ($2 \leq k \leq K$) elements in the memory module of a stack-based transition system. With the increase of K, the expressiveness of the corresponding system strictly increases. We propose an oracle deriving method which is guaranteed to find a sound transition sequence if one exits. Moreover, we introduce an effective approximation of that oracle, which decreases decoding ambiguity but practically covers almost exactly the same graphs for our data.

Based on the stronger transition system, we build a GR parser with a discriminative model for disambiguation and a beam decoder for inference. We conduct experiments on CTB 6.0 to profile this parser. With the increase of the K, the parser is able to utilize more GR graphs for training and the numeric performance is improved. Evaluation gauges how successful GR parsing for Chinese can be by applying data-driven models. Detailed analysis reveal some important factors that may possibly boost the performance. To our knowledge, this work provides the first result of extensive experiments of parsing Chinese with GRs.

We release our GR processing kit and gold-standard annotations for research purposes. These resources can be downloaded at `http://www.icst.pku.edu.cn/lcwm/omg`.

2 GB-grounded GR Extraction

In this section, we discuss the construction of the GR annotations. Basically, the annotations are automatically converted from a GB-grounded phrase-structure treebank, namely CTB. Conceptually, this conversion is similar to the conversions from CTB structures to representations in deep grammar formalisms (Tse and Curran, 2010; Yu et al., 2010; Guo et al., 2007; Xia, 2001). However, our work is grounded in GB, which is the linguistic basis of the construction of CTB. We argue that this theoretical choice makes the conversion process more compatible with the original annotations and therefore more accurate. We use directed graphs to explicitly encode bi-lexical dependencies involved in coordination, raising/control constructions, extraction, topicalization, and many other complicated phenomena. Fig. 1 shows an example of such a GR graph and its original CTB annotation.

2.1 Linguistic Basis

GRs are encoded in different ways in different languages. In some languages, e.g. Turkish, grammatical function is encoded by means of morphological marking, while in highly *configurational* languages, e.g. Chinese, the grammatical function of a phrase is heavily determined by its constituent structure position. Dominant Chomskyan theories, including GB, have defined GRs as configurations at phrase structures. Following this principle, CTB groups words into constituents through the use of a limited set of fundamental grammatical functions. Transformational grammar utilizes empty categories (ECs) to represent long-distance dependencies. In CTB, traces are provided by relating displaced linguistic material to where it should be interpreted semantically. By exploiting configurational information, traces and functional tag annotations, GR information can be hopefully

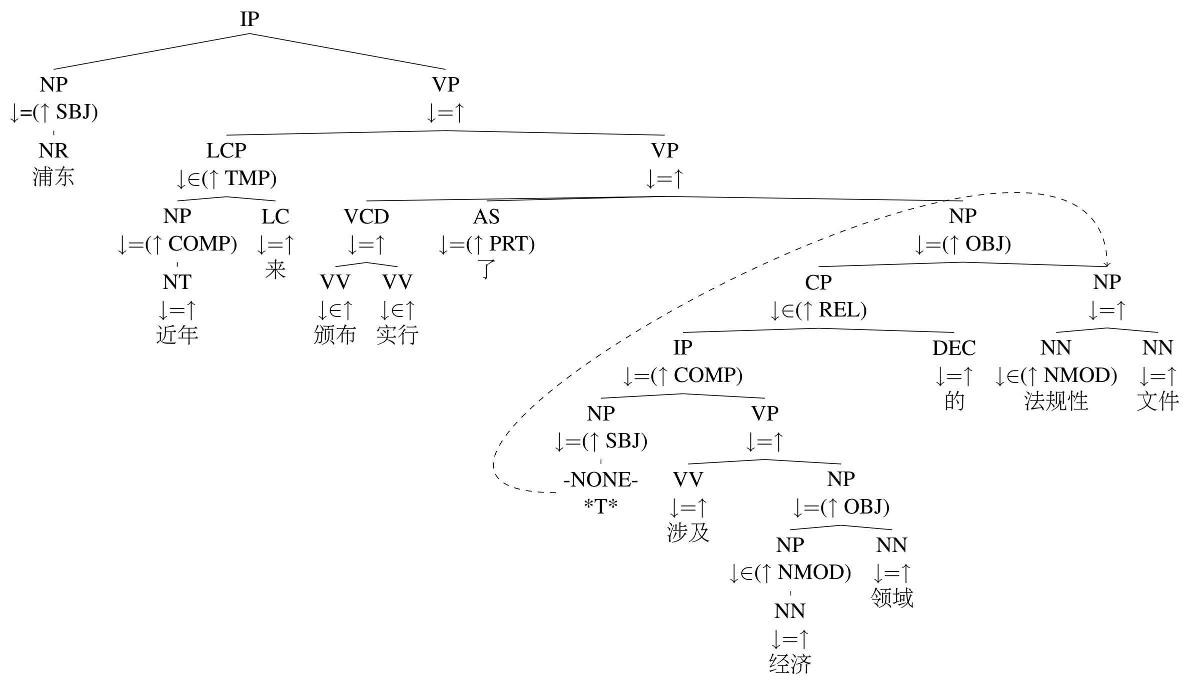

Figure 2: The original CTB annotation augmented with LFG-like f-structure annotations of the running example.

derived from CTB trees with high accuracy.

2.2 The Extraction Algorithm

Our treebank conversion algorithm borrows key insights from Lexical Functional Grammar (LFG; Bresnan and Kaplan, 1982; Dalrymple, 2001). LFG posits two levels of representation: c(onstituent)-structure and f(unctional)-structure minimally. C-structure is represented by phrase-structure trees, and captures surface syntactic configurations such as word order, while f-structure encodes grammatical functions. It is easy to extract a dependency backbone which approximates basic predicate-argument-adjunct structures from f-structures. The construction of the widely used PARC DepBank (King et al., 2003) is a good example.

LFG relates c-structure and f-structure through f-structure annotations, which compositionally map every constituent to a corresponding f-structure. Borrowing this key idea, we translate CTB trees to dependency graphs by first augmenting each constituency with f-structure annotations, then propagating the head words of the head or conjunct daughter(s) upwards to their parents, and finally creating a dependency graph. The following presents details step-by-step.

Tapping implicit information. Xue (2007) introduced a systematic study to tap the implicit functional information of CTB. This gives us a very good start to extract GRs. We slightly modify their method to enrich a CTB tree with f-structure annotations: Each node in a resulting tree is annotated with one and only one corresponding equation. See Fig. 2 for example. Comparing the original annotation and enriched one, we can see that the functionality of this step is to explicitly represent and regulate grammatical functions.

Beyond CTB annotations: tracing more. Natural languages do not always interpret linguistic material locally. In order to obtain accurate and complete GR, predicate-argument, or logical form representations, a hallmark of deep grammars is that they usually involve a non-local dependency resolution mechanism. CTB trees utilize ECs and coindexed materials to represent long-distance dependencies. An EC is a nominal element that does not have any phonological content and is therefore unpronounced. Two kinds of anaphoric ECs, i.e. big PRO and trace, are annotated in CTB. Theoretically speaking, only trace is generated as the result of *movement* and therefore annotated with antecedents in CTB. We carefully check the annotation and find that a considerable amount of antecedents are not labeled, and hence a lot of impor-

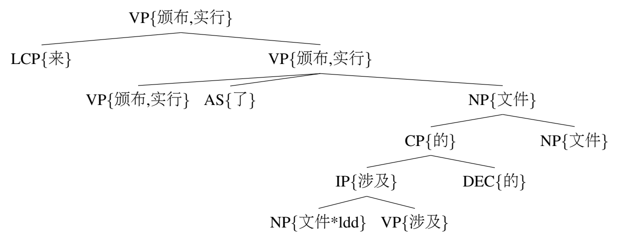

Figure 3: An example of lexicalized tree after head word upward passing. Only partial result is shown. The long-distance dependency between "涉及/involve" and "文件/document" is created through copying the dependent to a coindexed anaphoric EC position.

tant non-local information is missing. In addition, since the big PRO is also anaphoric, it is possible to find coindexed components sometimes. Such non-local information is also very valuable.

Beyond CTB annotations, we introduce a number of phrase-structure patterns to extract more non-local dependencies. The method heavily leverages linguistic rules to exploit structural information. We take into account both theoretical assumptions and analyzing practices to enrich coindexation information according to phrase-structure patterns. In particular, we try to link an anaphoric EC e with its c-commonders if no non-empty antecedent has already been coindexed with e. Because the CTB is influenced deeply by the X-bar syntax, which regulates constituent analysis much, the number of our linguistic rules is quite modest. For the development of conversion rules, we used the first 9 files of CTB, which contains about 100 sentences. Readers can refer to the well-documented Perl script for details. See Fig. 2 for example. The noun phrase "法规性文件/regulatory documents" is related to the trace "*T*." This coindexation is not labeled by the original annotation.

Passing head words and linking ECs. Based on an enriched tree, our algorithm propagates the head word of the head daughter upwards to their parents, linking coindexed units, and finally creating a GR graph. The partial result after head word passing of the running example is shown in Fig. 3. There are two differences of the head word passing between our GR extraction and a "normal" dependency tree extraction. First, the GR extraction procedure may pass multiple head words to its parent, especially in a coordination construction. Second, long-distance dependencies are created by linking ECs and their coindexed phrases.

	Precision	Recall	F-score
Unlabeled	99.48	99.17	99.32
Labeled	99.17	98.87	99.02

Table 1: Manual evaluation of 209 sentences.

2.3 Manual Evaluation

To have a precise understanding of whether our extraction algorithm works well, we have selected 20 files that contains 209 sentences in total for manual evaluation. Linguistic experts carefully examine the corresponding GR graphs derived by our extraction algorithm and correct all errors. In other words, a *gold standard* GR annotation set is created. The measure for comparing two dependency graphs is precision/recall of GR tokens which are defined as $\langle w_h, w_d, l \rangle$ tuples, where w_h is the head, w_d is the dependent and l is the relation. Labeled precision/recall (LP/LR) is the ratio of tuples correctly identified by the automatic generator, while unlabeled precision/recall (UP/UR) is the ratio regardless of l. F-score is a harmonic mean of precision and recall. These measures correspond to attachment scores (LAS/UAS) in dependency tree parsing. To evaluate our GR parsing models that will be introduced later, we also report these metrics.

The overall performance is summarized in Tab. 1. We can see that the automatical GR extraction achieves relatively high performance. There are two sources of errors in treebank conversion: (1) inadequate conversion rules and (2) wrong or inconsistent original annotations. During the creation of the gold standard corpus, we find that the former is mainly caused by complicated unbounded dependencies and the lack of internal structure for some kinds of phrases. Such problems are very hard to solve through rules only, if not possible, since original annotations do not provide sufficient information. The latter problem is more scattered and unpredictable.

2.4 Statistics

Allowing non-projective dependencies generally makes parsing either by graph-based or transition-based dependency parsing harder. Substantial research effort has been devoted in recent years to the design of elegant solutions for this problem. There are much more crossing arcs in the GR graphs than syntactic dependency trees. In the training data (defined in Section 4.1), there are 558132 arcs and 86534 crossing pairs, About half of the sentences have crossing arcs (10930 out of 22277). The wide existence of crossing arcs poses an essential challenge for GR parsing, namely, to find methods for handling crossing arcs without a significant loss in accuracy and efficiency.

3 Transition-based GR Parsing

The availability of large-scale treebanks has contributed to the blossoming of statistical approaches to build accurate *shallow* constituency and dependency parsers. With high-quality GR resources at hand, it is possible to study statistical approaches to automatically parse GR graphs. In this section, we investigate the feasibility of applying a data-driven, grammar-free approach to build GRs directly. In particular, transition-based dependency parsing method is studied.

3.1 Data-Driven Dependency Parsing

Data-driven, grammar-free dependency parsing has received an increasing amount of attention in the past decade. Such approaches, e.g. transition-based (Yamada and Matsumoto, 2003; Nivre, 2008) and graph-based (McDonald, 2006; Torres Martins et al., 2009) models have attracted the most attention of dependency parsing in recent years. Transition-based parsers utilize transition systems to derive dependency trees together with treebank-induced statistical models for predicting transitions. This approach was pioneered by (Yamada and Matsumoto, 2003) and (Nivre et al., 2004). Most research concentrated on surface syntactic structures, and the majority of existing approaches are limited to producing only trees. We notice two exceptions. Sagae and Tsujii (2008) and Titov et al. (2009) individually introduced two transition systems that can generate specific graphs rather than trees. Inspired by their work, we study transition-based approach to build GR graphs.

3.2 Transition Systems

Following (Nivre, 2008), we define a transition system for dependency parsing as a quadruple $S = (\mathcal{C}, T, c_s, \mathcal{C}_t)$, where

1. $\mathcal{C}$ is a set of configurations, each of which contains a buffer β of (remaining) words and a set A of dependency arcs,

Transitions	
Shift	$(\sigma, j \vert \beta, A) \Rightarrow (\sigma \vert j, \beta, A)$
Left-Arc$_l$	$(\sigma \vert i, j \vert \beta, A) \Rightarrow (\sigma \vert i, j \vert \beta, A \cup \{(j, l, i)\})$
Right-Arc$_l$	$(\sigma \vert i, j \vert \beta, A) \Rightarrow (\sigma \vert i, j \vert \beta, A \cup \{(i, l, j)\})$
Pop	$(\sigma \vert i, \beta, A) \Rightarrow (\sigma, \beta, A)$
Rotate$_k$	$(\sigma \vert i_k \vert \ldots \vert i_2 \vert i_1, \beta, A) \Rightarrow (\sigma \vert i_1 \vert i_k \vert \ldots \vert i_2, \beta, A)$

Table 2: K-permutation System.

2. T is a set of transitions, each of which is a (partial) function $t : \mathcal{C} \mapsto \mathcal{C}$,

3. c_s is an initialization function, mapping a sentence x to a configuration, with $\beta = [1, \ldots, n]$,

4. $\mathcal{C}_t \subseteq \mathcal{C}$ is a set of terminal configurations.

Given a sentence $x = w_1, \ldots, w_n$ and a graph $G = (V, A)$ on it, if there is a sequence of transitions $t_1, \ldots, t_m$ and a sequence of configurations $c_0, \ldots, c_m$ such that $c_0 = c_s(x)$, $t_i(c_{i-1}) = c_i (i = 1, \ldots, m)$, $c_m \in \mathcal{C}_t$, and $A_{c_m} = A$, we say the sequence of transitions is an *oracle* sequence. And we define $\bar{A}_{c_i} = A - A_{c_i}$ for the arcs to be built in c_i. In a typical transition-based parsing process, the input words are put into a queue and partially built structures are organized by a stack. A set of Shift/Reduce actions are performed sequentially to consume words from the queue and update the partial parsing results.

3.3 Online Reordering

Among existing systems, Sagae and Tsujii's is designed for projective graphs (denoted by $\mathcal{G}_1$ in Definition 1), and Titov et al.'s handles only a specific subset of non-projective graphs as well as projective graphs ($\mathcal{G}_2$). Applied to our data, only 51.0% and 76.5% of the extracted graphs are parsable with their systems. Obviously, it is necessary to investigate new transition systems for the parsing task in our study. To deal with crossing arcs, Titov et al. (2009) and Nivre (2009) designed a Swap transition that switches the position of the two topmost nodes on the stack. Inspired by their work, we extend this approach to parse more general graphs. The basic idea is to provide our new system with an ability to reorder more nodes during decoding in an online fashion, which we refer to as online reordering.

3.4 K-Permutation System

We define a K-permutation transition system $S_K = (\mathcal{C}, T, c_s, \mathcal{C}_t)$, where a configuration $c =$

$(\sigma, \beta, A) \in \mathcal{C}$ contains a stack σ of nodes besides β and A. We set the initial configuration for a sentence $x = w_1, \ldots, w_n$ to be $c_s(x) = ([], [1, \ldots, n], \{\})$, and take $\mathcal{C}_t$ to be the set of all configurations of the form $c_t = (\sigma, [], A)$ (for any arc set A). The set of transitions T contains five types of actions, as shown in Tab. 2:

1. SHIFT removes the front element from β and pushes it onto σ.
2. LEFT-ARC$_l$/RIGHT-ARC$_l$ updates a configuration by adding $(i, l, j)/(j, l, i)$ to A where i is the top of σ, and j is the front of β.
3. POP deletes the top element of σ.
4. ROTATE$_k$ updates a configuration with stack $\sigma|i_k|\ldots|i_2|i_1$ by rotating the top k nodes in stack left by one index, obtaining $\sigma|i_1|i_k|\ldots|i_2$, with constraint $2 \leq k \leq K$.

We refer to this system as K-permutation because by rotating the top k $(2 \leq k \leq K)$ nodes in the stack, we can obtain all the permutations of the top K nodes. Note that S_2 is identical to Titov et al.'s; S_∞ is complete with respect to the class of all directed graphs without self-loop, since we can arbitrarily permute the nodes in the stack. The K-permutation system exhibits a nice property: The sets of corresponding graphs are strictly monotonic with respect to the $\subset$ operation.

Definition 1. If a graph G can be parsed with transition system S_K, we say G is a *K-perm graph.* We use $\mathcal{G}_K$ to denote the set of all k-perm graphs. Specially, $\mathcal{G}_0 = \emptyset$, $\mathcal{G}_1$ is the set of all projective graphs, and $\mathcal{G}_\infty = \bigcup_{k=0}^{\infty} \mathcal{G}_k$.

Theorem 1. $\mathcal{G}_i \subsetneq \mathcal{G}_{i+1}, \forall i \geq 0.$

Proof. It is obvious that $\mathcal{G}_i \subseteq \mathcal{G}_{i+1}$ and $\mathcal{G}_0 \subsetneq \mathcal{G}_1$. Fig. 4 gives an example which is in $\mathcal{G}_{i+1}$ but not in $\mathcal{G}_i$ for all $i > 0$, indicating $\mathcal{G}_i \neq \mathcal{G}_{i+1}$. □

Theorem 2. $\mathcal{G}_\infty$ is the set of all graphs without self-loop.

Proof. It follows immediately from the fact that $G \in \mathcal{G}_{|V|}, \forall G = \langle V, E \rangle$. □

The transition systems introduced in (Sagae and Tsujii, 2008) and (Titov et al., 2009) can be viewed as S_1[1] and S_2.

[1]Though Sagae and Tsujii (2008) introduced additional constraints to exclude cyclic path, the fundamental transition mechanism of their system is the same to S_1.

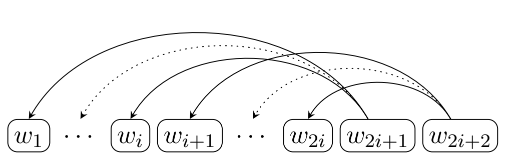

Figure 4: A graph which is in $\mathcal{G}_{i+1}$, but not in $\mathcal{G}_i$.

3.5 Normal Form Oracle

The K-permutation transition system may allow multiple oracle transition sequences on one graph, but trying to sum all the possible oracles is usually computational expensive. Here we give a construction procedure which is guaranteed to find an oracle sequence if one exits. We refer it as normal form oracle (NFO).

Let $L(j)$ be the ordered list of nodes connected to j in $\bar{A}_{c_{i-1}}$ for $j \in \sigma_{c_{i-1}}$, and let $\mathcal{L}_K(\sigma_{c_{i-1}}) = [L(j_1), \ldots, L(j_{\max\{l,K\}})]$. If $\sigma_{c_{i-1}}$ is empty, then we set t_i to SHIFT; if there is no arc linked to j_1 in $\bar{A}_{c_{i-1}}$, then we set t_i to POP; if there exits $a \in \bar{A}_{c_{i-1}}$ linking j_1 and b, then we set t_i to LEFT-ARC or RIGHT-ARC correspondingly. When there are only SHIFT and ROTATE left, we first apply a sequence of ROTATE's to make $\mathcal{L}_K(\sigma)$ complete ordered by lexicographical order, then apply a SHIFT. Let $c_i = t_i(c_{i-1})$, we continue to compute t_{i+1}, until β_{c_i} is empty.

Theorem 3. If a graph is parsable with the transition system S_K then the construction procedure is guaranteed to find an oracle transition sequence.

Proof. During the construction, all the arcs are built by LEFT-ARC or RIGHT-ARC, which links the top of the stack and the front of the buffer. Therefore, we prefer $\mathcal{L}(\sigma)$ to be as orderly as possible, to make the words to be linked sooner on the top of the stack. the construction procedure above does best within the power of the system S_K. □

3.6 An Approximation for NFO

In the construction of NFO transitions, we exhaustively use the ROTATE's to make $\mathcal{L}(\sigma)$ complete ordered. We also observed that the transition LEFT-ARC, RIGHT-ARC and SHIFT only change the relative order between the first element of $\mathcal{L}(\sigma)$ and the rest elements. Therefore we explored an approximate procedure to determine the ROTATE's, based on the observation. We call it approximate NFO (ANFO). Using notation defined in Section 3.5, the approximate procedure goes as follows. When it comes to the determination of

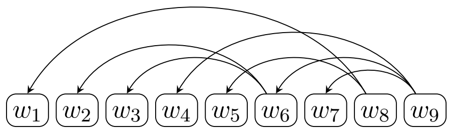

Figure 5: A graph that can be parsed with S_3 with a transition sequence SSSSR$_3$SR$_3$APAP**R**$_2$**R**$_3$SR$_3$SR$_3$APAPAPAPAP, where S stands for SHIFT, R for ROTATE, A for LEFT-ARC, and P for POP. But the approximate procedure fails to find the oracle, since **R**$_2$**R**$_3$ in bold in the sequence are not to be applied.

the ROTATE sequence, let k be the largest m such that $0 \leq m \leq \min\{K, l\}$ and $L(j_m)$ strictly precedes $L(j_1)$ by the lexicographical order (here we assume $L(j_0)$ strictly precedes any $L(j), j \in \sigma$). If $k > 0$, we set t_i to ROTATE$_k$; else we set t_i to SHIFT. The approximation assumes $\mathcal{L}(\sigma)$ is completely ordered except the first element, and insert the first element to its proper place each time.

Definition 2. We define $\hat{\mathcal{G}}_K$ as the graphs the oracle of which can be extracted by S_K with the approximation procedure.

It can be inferred similarly that Theorem 1 and Theorem 2 also hold for $\hat{\mathcal{G}}$'s. However, the $\hat{\mathcal{G}}_K$ is not equal to $\mathcal{G}_K$ in non-trivial cases.

Theorem 4. $\hat{\mathcal{G}}_i \subsetneq \mathcal{G}_i, \forall i \geq 3$.

Proof. It is trivial that $\hat{\mathcal{G}}_i \subseteq \mathcal{G}_i$. An example graph that is in $\mathcal{G}_3$ but not in $\hat{\mathcal{G}}_3$ is shown in Figure 5, examples for arbitrary $i > 3$ can be constructed similarly. □

The above theorem indicates the inadequacy of the ANFO deriving procedure. Nevertheless, empirical evaluation (Section 4.2) shows that the coverage of AFO and ANFO deriving procedures are almost identical when applying to linguistic data.

3.7 Statistical Parsing

When we parse a sentence $w_1 w_2 \cdots w_n$, we start with the initial configuration $c_0 = c_s(x)$, and choose next transition $t_i = C(c_{i-1})$ iteratively according to a discriminative classifier trained on oracle sequences. To build a parser, we use a structured classifier to approximate the oracle, and apply the Passive-Aggressive (PA) algorithm (Crammer et al., 2006) for parameter estimation. The PA algorithm is similar to the Perceptron algorithm, the difference from which is the update of weight vector. We also use parameter averaging and early update to achieve better training. Developing features has been shown crucial to advancing the state-of-the-art in dependency tree parsing (Koo and Collins, 2010; Zhang and Nivre, 2011). To build accurate deep dependency parsers, we utilize a large set of features for disambiguation. See the notes included in the supplymentary material for details. To improve the performance, we also apply the technique of beam search, which keep a beam of transition sequences with highest scores when parsing.

4 Experiments

4.1 Experimental setup

CTB is a segmented, part-of-speech (POS) tagged, and fully bracketed corpus in the constituency formalism, and very popular to evaluate fundamental NLP tasks, including word segmentation (Sun and Xu, 2011), POS tagging (Sun and Uszkoreit, 2012), and syntactic parsing (Zhang and Clark, 2009; Sun and Wan, 2013). We use CTB 6.0 and define the training, development and test sets according to the CoNLL 2009 shared task. We use gold-standard word segmentation and POS tagging results as inputs. All transition-based parsing models are trained with beam 16 and iteration 30. Overall precision/recall/f-score with respect to dependency tokens is reported. To evaluate the ability to recover non-local dependencies, the recall of such dependencies are reported too.

4.2 Coverage and Accuracy

There is a dual effect of the increase of the parameter k to our transition-based dependency parser. On one hand, the higher k is, the more expressivity the corresponding transition system has. A system with higher k covers more structures and allows to use more data for training. On the other hand, higher k brings more ambiguities to the corresponding parser, and the parsing performance may thus suffer. Note that the ambiguity exists not only in each step for transition decision, but also in selecting the training oracle.

The left-most columns of Tab. 3 shows the coverage of K-permutation transition system with respect to different K and different oracle deriving algorithms. Readers may be surprised that the coverage of NFO and ANFO deriving procedures is the same. Actually, all the covered graphs by the two oracle deriving procedures are exactly the

System	NFO	ANFO	UP	UR	UF	LP	LR	LF	UR$_L$	LR$_L$	UR$_{NL}$	LR$_{NL}$
S_2	76.5	76.5	85.88	81.00	83.37	83.98	79.21	81.53	81.93	80.34	58.88	52.17
S_3	89.0	89.0	86.02	81.72	83.82	84.07	79.86	81.91	82.61	80.94	60.46	54.28
S_4	95.6	95.6	86.28	82.06	84.12	84.35	80.22	82.23	82.92	81.29	61.48	54.77
S_5	98.4	98.4	86.44	82.21	84.27	84.51	80.37	82.39	83.15	81.51	59.80	53.30

Table 3: Coverage and accuracy of the GR parser on the development data.

same, except for S_3. Only 1 from 22277 sentences can find a NFO but not an ANFO. This number demonstrates the effectiveness of ANFO. In the following experiments, we use the ANFO's to train our parser.

Applied to our data, S_2, i.e. the exact system introduced by Titov et al. (2009), only covers 76.5% GR graphs. This is very different from the result obtained on the CoNLL shared task data for English semantic role labeling (SRL). According to (Titov et al., 2009), 99% semantic-role-labelled graphs can be generated by S_2. We think there are two main reasons accounting for the differences, and highlight the importance of the expressiveness of transition systems to solve deep dependency parsing problems. First, the SRL task only focuses on finding arguments and adjuncts of verbal (and nominal) predicates, while dependencies headed by other words are not contained in its graph representation. On contrast, a deep dependency structure, like GR graph, approximates deep syntactic or semantic information of a sentence as a whole, and therefore is more dense. As a result, permutation system with a very low k is incapable to handle more cases. Another reason is about the Chinese language. Some language-specific properties result in complex crossing arcs. For example, serial verb constructions are widely used in Chinese to describe several separate events without conjunctions. The verbal heads in such constructions share subjects and adjuncts, both of which are before the heads. The distributive dependencies between verbal heads and subjects/adjuncts usually produce crossing arcs (see Fig. 6). To test our assumption, we evaluate the coverage of S_2 over the functor-argument dependency graphs provided by the English and Chinese CCGBank (Hockenmaier and Steedman, 2007; Tse and Curran, 2010). The result is **96.9%** vs. **89.0%**, which confirms our linguistic intuition under another grammar formalism.

Tab. 3 summarizes the performance of the transition-based parser with different configurations to reveal how well data-driven parsing can be performed in realistic situations. We can see that with the increase of K, the overall parsing accuracy incrementally goes up. The high complexity of Chinese deep dependency structures demonstrates the importance of the expressiveness of a transition system, while the improved numeric accuracies practically certify the benefits. The two points merit further exploration to more expressive transition systems for deep dependency parsing, at least for Chinese. The labeled evaluation scores on the final test data are presented in Tab. 4.

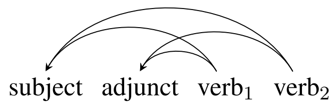

Figure 6: A simplified example to illustrate crossing arcs in serial verbal constructions.

Test	UP	UR	UF	LR$_L$	LR$_{NL}$
S_5	83.93	79.82	81.82	80.94	54.38

Table 4: Performance on the test data.

4.3 Precision vs. Recall

A noteworthy thing about the overall performance is that the precision is promising but the recall is too low behind. This difference is consistent with the result obtained by a shift-reduce CCG parser (Zhang and Clark, 2011). The functor-argument dependencies generated by that parser also has a relatively high precision but considerably low recall. There are two similarities between our parser and theirs: 1) both parsers produce dependency graphs rather trees; 2) both parser employ a beam decoder that does not guarantee global optimality. To build NLP application, e.g. information extraction, systems upon GR parsing, such property merits attention. A good trade-off between the precision and the recall may have a great impact on final results.

4.4 Local vs. Non-local

Although the micro accuracy of all dependencies are considerably good, the ability of current state-of-the-art statistical parsers to find difficult non-local materials is far from satisfactory, even for English (Rimell et al., 2009; Bender et al., 2011). We report the accuracy in terms of local and non-local dependencies respectively to show the difficulty of the recovery of non-local dependencies. The last four columns of Tab. 3 demonstrates the labeled/unlabeled recall of local (UR_L/LR_L) and non-local dependencies (UR_NL/LR_NL). We can clearly see that non-local dependency recovery is extremely difficult for Chinese parsing.

4.5 Deep vs. Deep

CCG and HPSG parsers also favor the dependency-based metrics for evaluation (Clark and Curran, 2007b; Miyao and Tsujii, 2008). Previous work on Chinese CCG and HPSG parsing unanimously agrees that obtaining the deep analysis of Chinese is more challenging (Yu et al., 2011; Tse and Curran, 2012). The successful C&C and Enju parsers provide very inaccurate results for Chinese texts. Though the numbers profiling the qualities of deep dependency structures under different formalisms are not directly comparable, all empirical evaluation indicates that the state-of-the-art of deep linguistic processing for Chinese lag behind very much.

5 Related Work

Wide-coverage in-depth and accurate linguistic processing is desirable for many practical NLP applications, such as machine translation (Wu et al., 2010) and information extraction (Miyao et al., 2008). Parsing in deep formalisms, e.g. CCG, HPSG, LFG and TAG, provides valuable, richer linguistic information, and researchers thus draw more and more attention to it. Very recently, study on deep linguistic processing for Chinese has been initialized. Our work is one of them.

To quickly construct deep annotations, corpus-driven grammar engineering has been studied. Phrase structure trees in CTB have been semi-automatically converted to deep derivations in the CCG (Tse and Curran, 2010), LFG (Guo et al., 2007), TAG (Xia, 2001) and HPSG (Yu et al., 2010) formalisms. Our GR extraction work is similar, but grounded in GB, which is more consistent with the construction of the original annotations.

Based on converted fine-grained linguistic annotations, successful English deep parsers, such as C&C (Clark and Curran, 2007b) and Enju (Miyao and Tsujii, 2008), have been evaluated (Yu et al., 2011; Tse and Curran, 2012). We also borrow many ideas from recent advances in deep syntactic or semantic parsing for English. In particular, Sagae and Tsujii (2008)'s and Titov et al. (2009)'s studies on transition-based deep dependency parsing motivated our work very much. However, simple adoption of their systems does not resolve Chinese GR parsing well because the GR graphs are much more complicated. Our investigation on the K-permutation transition system advances the capacity of existing methods.

6 Conclusion

Recent years witnessed rapid progress made on deep linguistic processing for English, and initial attempts for Chinese. Our work stands in between traditional dependency tree parsing and deep linguistic processing. We introduced a system for automatically extracting grammatical relations of Chinese sentences from GB phrase structure trees. The present work remedies the resource gap by facilitating the accurate extraction of GR annotations from GB trees. Manual evaluation demonstrate the effectiveness of our method. With the availability of high-quality GR resources, transition-based methods for GR parsing was studied. A new formal system, namely K-permutation system, is well theoretically discussed and practically implemented as the core module of a deep dependency parser. Empirical evaluation and analysis were presented to give better understanding of the Chinese GR parsing problem. Detailed analysis reveals some important directions for future investigation.

Acknowledgement

The work was supported by NSFC (61300064, 61170166 and 61331011) and National High-Tech R&D Program (2012AA011101).

References

Emily M. Bender, Dan Flickinger, Stephan Oepen, and Yi Zhang. 2011. Parser evaluation over local and non-local deep dependencies in a large corpus. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 397–408. Association for Computational Linguistics, Edinburgh, Scotland, UK. URL http://www.aclweb.org/anthology/D11-1037.

J. Bresnan and R. M. Kaplan. 1982. Introduction: Grammars as mental representations of language. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages xvii–lii. MIT Press, Cambridge, MA.

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the parc depbank. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 41–48. Association for Computational Linguistics, Sydney, Australia. URL http://www.aclweb.org/anthology/P/P06/P06-2006.

Andrew Carnie. 2007. *Syntax: A Generative Introduction*. Blackwell Publishing, Blackwell Publishing 350 Main Street, Malden, MA 02148-5020, USA, second edition.

Noam Chomsky. 1981. *Lectures on Government and Binding*. Foris Publications, Dordecht.

Stephen Clark and James Curran. 2007a. Formalism-independent parser evaluation with ccg and depbank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 248–255. Association for Computational Linguistics, Prague, Czech Republic. URL http://www.aclweb.org/anthology/P07-1032.

Stephen Clark and James R. Curran. 2007b. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Comput. Linguist.*, 33(4):493–552. URL http://dx.doi.org/10.1162/coli.2007.33.4.493.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *JOURNAL OF MACHINE LEARNING RESEARCH*, 7:551–585.

M. Dalrymple. 2001. *Lexical-Functional Grammar*, volume 34 of *Syntax and Semantics*. Academic Press, New York.

Yuqing Guo, Josef van Genabith, and Haifeng Wang. 2007. Treebank-based acquisition of lfg resources for Chinese. In *Proceedings of the LFG07 Conference*. CSLI Publications, California, USA.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396.

Ron Kaplan, Stefan Riezler, Tracy H King, John T Maxwell III, Alex Vasserman, and Richard Crouch. 2004. Speed and accuracy in shallow and deep stochastic parsing. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 97–104. Association for Computational Linguistics, Boston, Massachusetts, USA.

Tracy Holloway King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 dependency bank. In *In Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*, pages 1–8.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11. Association for Computational Linguistics, Uppsala, Sweden. URL http://www.aclweb.org/anthology/P10-1001.

Ryan McDonald. 2006. *Discriminative learning and spanning tree algorithms for dependency parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA.

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proceedings of ACL-08: HLT*, pages 46–54. Association for Computational Linguistics, Columbus, Ohio. URL http://www.aclweb.org/anthology/P/P08/P08-1006.

Yusuke Miyao, Kenji Sagae, and Jun'ichi Tsujii. 2007. Towards framework-independent evaluation of deep linguistic parsers. In Ann Copestake, editor, *Proceedings of the GEAF 2007 Workshop*, CSLI Studies in Computational Linguistics Online, page 21 pages. CSLI Publications. URL http://www.cs.cmu.edu/~sagae/docs/geaf07miyaoetal.pdf.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic hpsg parsing. *Comput. Linguist.*, 34(1):35–80. URL http://dx.doi.org/10.1162/coli.2008.34.1.35.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.*, 34:513–553. URL http://dx.doi.org/10.1162/coli.07-056-R1-07-027.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359. Association for Computational Linguistics, Suntec, Singapore. URL http://www.aclweb.org/anthology/P/P09/P09-1040.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In Hwee Tou Ng and Ellen Riloff, editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 49–56. Association for Computational Linguistics, Boston, Massachusetts, USA.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 813–821. Association for Computational Linguistics, Singapore. URL http://www.aclweb.org/anthology/D/D09/D09-1085.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 753–760. Coling 2008 Organizing Committee, Manchester, UK. URL http://www.aclweb.org/anthology/C08-1095.

Weiwei Sun and Hans Uszkoreit. 2012. Capturing paradigmatic and syntagmatic lexical relations: Towards accurate Chinese part-of-speech tagging. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Weiwei Sun and Xiaojun Wan. 2013. Data-driven, pcfg-based and pseudo-pcfg-based models for Chinese dependency parsing. *Transactions of the Association for Computational Linguistics (TACL)*.

Weiwei Sun and Jia Xu. 2011. Enhancing Chinese word segmentation using unlabeled data. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 970–979. Association for Computational Linguistics, Edinburgh, Scotland, UK. URL http://www.aclweb.org/anthology/D11-1090.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1562–1567. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. URL http://dl.acm.org/citation.cfm?id=1661445.1661696.

Andre Torres Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 342–350. Association for Computational Linguistics, Suntec, Singapore. URL http://www.aclweb.org/anthology/P/P09/P09-1039.

Daniel Tse and James R. Curran. 2010. Chinese CCGbank: extracting CCG derivations from the penn Chinese treebank. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1083–1091. Coling 2010 Organizing Committee, Beijing, China. URL http://www.aclweb.org/anthology/C10-1122.

Daniel Tse and James R. Curran. 2012. The challenges of parsing Chinese with combinatory categorial grammar. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 295–304. Association for Computational Linguistics, Montréal, Canada. URL http://www.aclweb.org/anthology/N12-1030.

Xianchao Wu, Takuya Matsuzaki, and Jun'ichi Tsujii. 2010. Fine-grained tree-to-string translation rule extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 325–334. Association for Computational Linguistics, Uppsala, Sweden. URL http://www.aclweb.org/anthology/P10-1034.

Fei Xia. 2001. *Automatic grammar generation from two different perspectives*. Ph.D. thesis, University of Pennsylvania.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The penn Chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11:207–238. URL http://portal.acm.org/citation.cfm?id=1064781.1064785.

Nianwen Xue. 2007. Tapping the implicit information for the PS to DS conversion of the Chinese treebank. In *Proceedings of the Sixth International Workshop on Treebanks and Linguistics Theories*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *The 8th International Workshop of Parsing Technologies (IWPT2003)*, pages 195–206.

Kun Yu, Yusuke Miyao, Takuya Matsuzaki, Xiangli Wang, and Junichi Tsujii. 2011. Analysis of the difficulties in Chinese deep parsing. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 48–57. Association for Computational Linguistics, Dublin, Ireland. URL http://www.aclweb.org/anthology/W11-2907.

Kun Yu, Miyao Yusuke, Xiangli Wang, Takuya Matsuzaki, and Junichi Tsujii. 2010. Semi-automatically developing Chinese hpsg grammar from the penn Chinese treebank for deep parsing. In *Coling 2010: Posters*, pages 1417–1425. Coling 2010 Organizing Committee, Beijing, China. URL http://www.aclweb.org/anthology/C10-2162.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the Chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 162–171. Association for Computational Linguistics, Paris, France. URL http://www.aclweb.org/anthology/W09-3825.

Yue Zhang and Stephen Clark. 2011. Shift-reduce CCG parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 683–692. Association for Computational Linguistics, Portland, Oregon, USA. URL http://www.aclweb.org/anthology/P11-1069.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193. Association for Computational Linguistics, Portland, Oregon, USA. URL http://www.aclweb.org/anthology/P11-2033.