A Statistical Constraint Dependency Grammar (CDG) Parser

Wen Wang
Speech Technology and Research Lab
SRI International
Menlo Park, CA 94025,
U.S.A.,
wwang@speech.sri.com

Mary P. Harper
Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285,
U.S.A.,
harper@ecn.purdue.edu

Abstract

CDG represents a sentence's grammatical structure as assignments of dependency relations to functional variables associated with each word in the sentence. In this paper, we describe a statistical CDG (SCDG) parser that performs parsing incrementally and evaluate it on the Wall Street Journal Penn Treebank. Using a tight integration of multiple knowledge sources, together with distance modeling and synergistic dependencies, this parser achieves a parsing accuracy comparable to several state-of-the-art context-free grammar (CFG) based statistical parsers using a dependency-based evaluation metric. Factors contributing to the SCDG parser's performance are analyzed.

1 Introduction

Statistical parsing has been an important focus of recent research (Magerman, 1995; Eisner, 1996; Charniak, 1997; Collins, 1999; Ratnaparkhi, 1999; Charniak, 2000). Several of these parsers generate constituents by conditioning probabilities on non-terminal labels, part-of-speech (POS) tags, and some headword information (Collins, 1999; Ratnaparkhi, 1999; Charniak, 2000). They utilize non-terminals that go beyond the level of a single word and do not explicitly use lexical features. Collins' Model 2 parser (1999) learns the distinction between complements and adjuncts by using heuristics during training, distinguishes complement and adjunct non-terminals, and includes a probabilistic choice of left and right subcategorization frames, while his Model 3 parser uses gap features to model wh-movement. Charniak (Charniak, 2000) developed a state-of-the-art statistical CFG parser and then built an effective language model based on it (Charniak, 2001). But his parser and language model were originally designed to analyze complete sentences. Among the statistical dependency grammar parsers, Eisner's (1996) best probabilistic dependency model used unlabeled links between words and their heads, as well as between words and their complements and adjuncts. However, the parser does not distinguish between complements and adjuncts or model wh-movement. Collins' bilexical dependency grammar parser (1999) used head-modifier relations between pairs of words much as in a dependency grammar, but they are limited to relationships between words in reduced sentences with base NPs.

Our research interest focuses on building a high quality statistical parser for language modeling. We chose CDG as the underlying grammar for several reasons. Since CDGs can be lexicalized at the word-level, a CDG parser-based language model is an important alternative to CFG parser-based models, which must model both non-terminals and terminals. Furthermore, the lexicalization of CDG parse rules is able to include not only lexical category information, but also a rich set of lexical features to model subcategorization and wh-movement. By using CDG, our statistical model is able to distinguish between adjuncts and complements. Additionally, CDG is more powerful than CFG and is able to model languages with crossing dependencies and free word ordering.

In this paper, we describe and evaluate a statistical CDG parser for which the probabilities of parse prefix hypotheses are incrementally updated when the next input word is available, i.e., it parses incrementally. Section 2 describes how CDG represents a sentence's parse and then defines a SuperARV, which is a lexicalization of CDG parse rules used in our parsing model. Section 3 presents the parsing model, while Section 4 motivates the evaluation metric used to evaluate our parser. Section 5 presents and discusses the experimental results.

2 CDG Parsing

CDG (Harper and Helzerman, 1995) represents syntactic structures using labeled dependencies between words. Consider an example CDG parse for the sentence *What did you learn* depicted in the white box of Figure 1. Each word in the parse has a lexical category, a set of feature values, and a set of roles that are assigned role values, each comprised of a label indicating the grammatical role of the word and its modifiee (i.e., the position of the word it is modifying when it takes on that role). Consider the role value assigned to the governor role (denoted G) of *you*, `np-2`. The label `np` indicates the grammatical function of *you* when it is governed by its head in position `2`. Every word in a sentence must have a governor role with an assigned role value. Need roles are used to ensure that the grammatical requirements of a word are met (e.g., subcategorization).

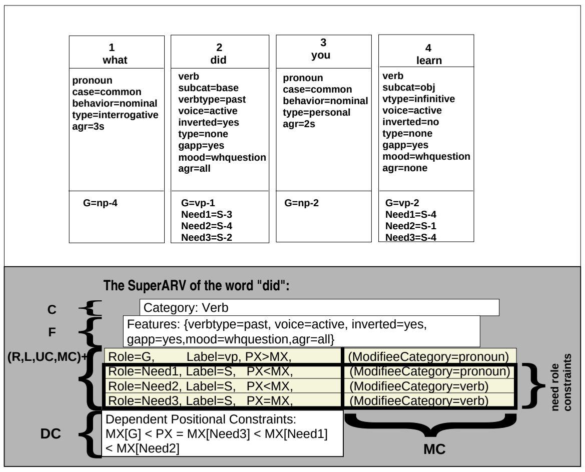

Figure 1: An example of a CDG parse and the SuperARV of the word *did* in the sentence *what did you learn*. PX and MX([R]) represent the position of a word and its modifi ee (for role R), respectively.

Note that CDG parse information can be easily lexicalized at the word level. This lexicalization is able to include not only lexical category and syntactic constraints, but also a rich set of lexical features to model subcategorization and wh-movement without a combinatorial explosion of the parametric space (Wang and Harper, 2002). CDG can distinguish between adjuncts and complements due to the use of need roles (Harper and Helzerman, 1995), is more powerful than CFG, and has the ability to model languages with crossing dependencies and free word ordering (hence, this research could be applicable to a wide variety of languages).

An almost-parsing LM based on CDG has been developed in (Wang and Harper, 2002). The underlying hidden event of this LM is a *SuperARV*. A SuperARV is formally defined as a four-tuple for a word, $\langle C, F, (R, L, UC, MC)+, DC \rangle$, where C is the lexical category of the word, $F = \{Fname_1 = Fvalue_1, \ldots, FName_f = FValue_f\}$ is a feature vector (where $Fname_i$ is the name of a feature and $Fvalue_i$ is its corresponding value), DC represents the relative ordering of the positions of a word and all of its modifiees, (R, L, UC, MC)+ is a list of one or more four-tuples, each representing an abstraction of a role value assignment, where R is a role variable, L is a functionality label, UC represents the relative position relation of a word and its dependent, and MC encodes some modifiee constraints, namely, the lexical category of the modifiee for this dependency relation. The gray box of Figure 1 presents an example of a SuperARV for the word *did*. From this example, it is easy to see that a SuperARV is a *join* on the role value assignments of a word, with explicit position information replaced by a relation that expresses whether the modifiee points to the current word, a previous word, or a subsequent word. The SuperARV structure provides an explicit way to organize information concerning one consistent set of dependency links for a word that can be directly derived from a CDG parse. SuperARVs encode lexical information as well as syntactic and semantic constraints in a uniform representation that is much more fine-grained than part-of-speech (POS). A sentence tagged with SuperARVs is an almost-parse since all that remains is to specify the precise position of each modifiee. SuperARV LMs have been effective at reducing word error rate (WER) on wide variety of continuous speech recognition (CSR) tasks, including Wall Street Journal (Wang and Harper, 2002), Broadcast News (Wang et al., 2003), and Switchboard tasks (Wang et al., 2004).

3 SCDG Parser

3.1 The Basic Parsing Algorithm

Our SCDG parser is a probabilistic generative model. It can be viewed as consisting of two components: SuperARV tagging and modifiee determination. These two steps can be either loosely or tightly integrated. To simplify discussion, we describe the loosely integrated version, but we implement and evaluate both strategies. The basic parsing algorithm for the loosely integrated case is summarized in Figure 2, with the algorithm's symbols defined in Table 1. In the first step, the top N-best SuperARV assignments are generated for an input sentence $w_1, \ldots, w_n$ using token-passing (Young et al., 1997) on a Hidden Markov Model with trigram probabilistic estimations for both transition and emission probabilities. Each SuperARV sequence for the sentence is represented as a sequence of tuples: $\langle w_1, s_1 \rangle, \ldots, \langle w_n, s_n \rangle$, where $\langle w_k, s_k \rangle$ represents the word w_k and its SuperARV assignment s_k. These assignments are stored in a stack ranked in non-increasing order by tag assignment probability.

During the second step, the modifiees are statistically specified in a left-to-right manner. Note that the algorithm utilizes modifiee lexical category constraints to filter out candidates with mismatched lexical categories. When processing the word $w_k, k = 1, \ldots, n$, the algorithm attempts to determine the left dependents of w_k from the closest to the farthest. The dependency assignment probability when choosing the $(c+1)^{\text{th}}$ left dependent (with its position denoted $dep(k, -(c+1))$) is defined as:

$$Pr(\text{link}(s_{dep(k,-(c+1))}, s_k, -(c+1)|syn, \mathcal{H}))$$

where $\mathcal{H} = \langle w, s\rangle_k, \langle w, s\rangle_{dep(k,-(c+1))}, \langle w, s\rangle_{dep(k,-1)}^{dep(k,-c)}$. The dependency assignment probability is conditioned on the word identity and SuperARV assignment of w_k and $w_{dep(k,-(c+1))}$ as well as all of the c previously chosen left dependents $\langle w, s\rangle_{dep(k,-1)}^{dep(k,-c)}$ for w_k. A Boolean random variable *syn* is used to model the synergistic relationship between certain role pairs. This mechanism allows us to elevate, for example, the probability that the subject of a sentence w_i is governed by a tensed verb w_j when the need role value of w_j points to w_i as its subject. The syn value for a dependency relation is determined heuristically based on the lexical category, role name, and label information of the two dependent words. After the algorithm statistically specifies the left dependents for w_k, it must also determine whether w_k could be the $(d+1)^{\text{th}}$ right dependent of a previously seen word $w_p, p = 1, \ldots, k-1$ (where d denotes the number of already assigned right dependents of w_p), as shown in Figure 2.

After processing word w_k in each partial parse on the stack, the partial parses are re-ranked according to their updated probabilities. This procedure is iterated until the top parse in the stack covers the entire sentence. For the tightly coupled parser, the SuperARV assignment to a word and specification of its modifiees are integrated into a single step. The parsing procedure, which is completely incremental, is implemented as a simple best-first stack-based search. To control time and memory complexity, we used two pruning thresholds: maximum stack depth and maximum difference between the log probabilities of the top and bottom partial parses in the stack. These pruning thresholds are tuned based on the tradeoff of time/memory complexity and parsing accuracy on a heldout set, and they both have hard limits.

Note the maximum likelihood estimation of dependency assignment probabilities in the basic loosely coupled parsing algorithm presented in Figure 2 is likely to suffer from data sparsity, and the estimates for the tightly coupled algorithm are likely to suffer even more so. Hence, we smooth the probabilities using Jelinek-Mercer smoothing (Jelinek, 1997), as described in (Wang and Harper, 2003; Wang, 2003).

3.2 Additions to the Basic Model

Some additional features are added to the basic model because of their potential to improve SCDG parsing accuracy. Their efficacy is evaluated in Section 5.

Modeling crossing dependencies: The basic parsing algorithm was implemented to preclude crossing dependencies; however, it is important to allow them in order to model wh-movement in some cases (e.g., wh-PPs).

Distance and barriers between dependents: Because distance between two dependent words is an important factor in determining the modifiees of a word, we evaluate an alternative model that adds distance, $\Delta_{dep(k,\pm(c+1)),k}$ to $\mathcal{H}$ in Figure 2. Note that $\Delta_{dep(k,\pm(c+1)),k}$ represents the distance between position $dep(k, \pm(c+1))$ and k. To avoid data sparsity problems, distance is bucketed and a discrete random variable is used to model it. We also model punctuation and verbs based on prior work. Like (Collins, 1999), we also found that verbs appear to act as barriers that impact modifiee links. Hence, a Boolean random variable that represents whether there is a verb between the dependencies is added to condition the probability estimations. Punctuation is treated similarly to coordination constructions with punctuation governed by the headword of the following phrase, and heuristic questions on punctuation were used to provide additional constraints on dependency assignments (Wang, 2003).

Modifiee lexical features: The SuperARV structure employed in the SuperARV LM (Wang and Harper, 2002) uses only lexical categories of modifiees as modifiee constraints. In previous work (Harper et al., 2001), modifiee lexical features were central to increasing the selectivity of a CDG. Hence, we have developed methods to add additional relevant lexical features to modifiee constraints of a SuperARV structure (Wang, 2003).

4 Parsing Evaluation Metric

To evaluate our parser, which generates CDG analyses rather than CFG constituent bracketing, we

Table 1: Definitions of symbols used in the basic parsing algorithm.

Term	**Denotes**
$\mathcal{L}(s_k)$, $\mathcal{R}(s_k)$	all dependents of s_k to the left and right of w_k, respectively
$N(\mathcal{L}(s_k))$, $N(\mathcal{R}(s_k))$	the number of left and right dependents of s_k, respectively
$dep(k, -c)$, $dep(k, c)$	c^{th} left dependent and right dependent of s_k, respectively
$dep(k, -1)$, $dep(k, 1)$	the position of the closest left dependent and right dependent of s_k, respectively
$dep(k, -N(\mathcal{L}(s_k)))$, $dep(k, N(\mathcal{L}(s_k)))$	the position of the farthest left dependent and right dependent of s_k, respectively
Cat(s_k)	the lexical category of s_k
ModCat($s_k, -c$), ModCat(s_k, c)	the lexical category of s_k's c^{th} left and right dependent (encoded in the SuperARV structure), respectively
link(s_i, s_j, k)	the dependency relation between SuperARV s_i and s_j with w_i assigned as the k^{th} dependent of s_j, e.g., link($s_{dep(k,-(c+1))}, s_k, -(c+1)$) indicates that $w_{dep(k,-(c+1))}$ is the $(c+1)^{\text{th}}$ left dependent of s_k.
$D(\mathcal{L}(s_k))$, $D(\mathcal{R}(s_k)))$	the number of left and right dependents of s_k already assigned, respectively
$\langle w, s \rangle_{dep(k,-1)}^{dep(k,-c)}$	words and SuperARVs of s_k's closest left dependent up to its c^{th} left dependent
$\langle w, s \rangle_{dep(k,1)}^{dep(k,c)}$	words and SuperARVs of s_k's closest right dependent up to its c^{th} right dependent
syn	a random variable denoting the synergistic relation between some dependents

can either convert the CDG parses to CFG bracketing and then use PARSEVAL, or convert the CFG bracketing generated from the gold standard CFG parses to CDG parses and then use a metric based on dependency links. Since our parser is trained using a CFG-to-CDG transformer (Wang, 2003), which maps a CFG parse tree to a unique CDG parse, it is sensible to evaluate our parser's accuracy using gold standard CDG parse relations. Furthermore, in the 1998 Johns Hopkins Summer workshop final report (Hajic et al., 1998), Collins et al. pointed out that in general the mapping from dependencies to tree structures is one-to-many: there are many possible trees that can be generated for a given dependency structure since, although generally trees in the Penn Treebank corpus are quite flat, they are not consistently "flat." This variability adds a non-deterministic aspect to the mapping from CDG dependencies to CFG parse trees that could cause spurious PARSEVAL scoring errors. Additionally, when there are crossing dependencies, then no tree can be generated for that set of dependencies. Consequently, we have opted to use a transformer to convert CFG trees to CDG parses and define a new dependency-based metric adapted from (Eisner, 1996). We define **role value labeled precision (RLP)** and **role value labeled recall (RLR)** on dependency links as follows:

$$RLP = \frac{\text{correct modifiee assignments}}{\text{number of modifiees our parser found}}$$

$$RLR = \frac{\text{correct modifiee assignments}}{\text{number of modifiess in the gold test set parses}}$$

where a correct modifiee assignment for a word w_i in a sentence means that a three-tuple $\langle$**role id**, **role label**, **modifiee word position**$\rangle$ (i.e., a role value) for w_i is the same as the three-tuple role value for the corresponding **role id** of w_i in the gold test parse. This differs from Eisner's (1996) precision and recall metrics which use no label information and score only parent (governor) assignments, as in traditional dependency grammars. We will evaluate role value labeled precision and recall on all roles of the parse, as well as the governor-only portion of a parse. Eisner (Eisner, 1996) and Lin (Lin, 1995) argued that dependency link evaluation metrics are valuable for comparing parsers since they are less sensitive than PARSEVAL to single misattachment errors that may cause significant error propagation to other constituents. This, together with the fact that we must train our parser using CDG parses generated in a lossy manner from a CFG treebank, we chose to use RLP and RLR to compare our parsing accuracy with several state-of-the-art parsers.

5 Evaluation and Discussion

All of the evaluations were performed on the Wall Street Journal Penn Treebank task. Following the traditional data setup, sections 02-21 are used for training our parser, section 23 is used for testing, and section 24 is used as the development set for parameter tuning and debugging. As in (Ratnaparkhi, 1999; Charniak, 2000; Collins, 1999), we evaluate on all sentences with length ≤ 40 words (2,245 sentences) and length ≤ 100 words (2,416 sentences). For training our probabilistic CDG parser on this task, the CFG bracketing of the training set is trans-

```
BASIC PARSING ALGORITHM
1. Using SuperARV tagging on word sequence $w_1, \ldots, w_n$, obtain a set of N-best SuperARV sequences with each
   element consisting of $n$ (word, SuperARV) tuples, denoted $\langle w_1, s_1 \rangle, \ldots, \langle w_n, s_n \rangle$, which we will call an assignment.
2. For each SuperARV assignment, initialize the stack of parse prefixes with this assignment:
   $/*$ From left-to-right, process each $\langle word, tag \rangle$ of the assignment and generate parse prefixes $*/$
   for $k :\, = 1, n$ do
   $/*$ Step a: $*/$
       /* decide left dependents of $\langle w_k, s_k \rangle$ from the nearest to the farthest */
       for $c$ from 0 to $N(\mathcal{L}(s_k)) - 1$ do
           $/*$ Choose a position for the $(c+1)^{\text{th}}$ left dependent of $\langle w_k, s_k \rangle$ from the set of possible positions
               $\mathcal{C} = \{1, \ldots, dep(k, -c) - 1\}$. The position choice is denoted $dep(k, -(c+1)) * /$
               $/*$ In the following equations, different left dependent assignments will generate
                   different parse prefixes, each of which is stored in the stack $* /$
               for each $dep(k, -(c+1))$ from positions $\mathcal{C} = \{1, \ldots, dep(k, -c) - 1\}$
                       $/*$ Check whether the lexical category of the choice matches the modifiee lexical
                           category of the $(c+1)^{\text{th}}$ left dependent of $\langle w_k, s_k \rangle * /$
                       if Cat$(s_{dep(k,-(c+1))})$ == ModCat$(s_k, -(c+1))$ then
                           $Pr(T) :\, = Pr(T) \times Pr(\text{link}(s_{dep(k,-(c+1))}, s_k, -(c+1) | syn, \mathcal{H}))$
                           where $\mathcal{H} = \langle w, s \rangle_k, \langle w, s \rangle_{dep(k,-(c+1))}, \langle w, s \rangle_{dep(k,-1)}^{dep(k,-c)}$
           $/*$ End of choosing left dependents of $\langle w_k, s_k \rangle$ for this parse prefix $*/$
   $/*$ Step b: $*/$
           $/*$ For the word/tag pair $\langle w_k, s_k \rangle$, check whether it could be a right dependent of any previously
               seen word within a parse prefix of $\langle w_1, s_1 \rangle, \ldots, \langle w_{k-1}, s_{k-1} \rangle$ */
           for $p :\, = 1, k-1$ do
               $/*$ If $\langle w_p, s_p \rangle$ still has right dependents left unspecified, then try out $\langle w_k, s_k \rangle$ as a right dependent */
               if $D(\mathcal{R}(s_p)) \neq N(\mathcal{R}(s_p))$ then
                   $d :\, = D(\mathcal{R}(s_p))$
                   $/*$ If the lexical category of $\langle w_k, s_k \rangle$ matches the modifiee lexical category of the $(d+1)^{\text{th}}$ right
                       dependent of $\langle w_p, s_p \rangle$, then $s_k$ might be $\langle w_p, s_p \rangle$'s $(d+1)^{\text{th}}$ right dependent $* /$
                   if Cat$(s_k)$ == ModCat$(s_p, d+1)$ then
                       $Pr(T) :\, = Pr(T) \times Pr(\text{link}(s_k, s_p, d+1) | syn, \mathcal{H})$, where $\mathcal{H} = \langle w, s \rangle_p, \langle w, s \rangle_k, \langle w, s \rangle_{dep(p,1)}^{dep(p,d)}$
       Sort the parse prefixes in the stack according to $logPr(T)$ and apply pruning using the thresholds.
3. After processing $w_1, \ldots, w_n$, pick the parse with the highest $logPr(T)$ in the stack as the parse for that sentence.
```

Figure 2: The basic loosely coupled parsing algorithm. Note the algorithm updates the probabilities of parse prefix hypotheses incrementally when processing each input word.

formed into CDG annotations using a CFG-to-CDG transformer (Wang, 2003). Note that the soundness of the CFG-to-CDG transformer was evaluated by examining the CDG parses generated from the transformer on the Penn Treebank development set to ensure that they were correct given our grammar definition.

5.1 Contribution of Model Factors

First, we investigate the contribution of the model additions described in Section 3 to parse accuracy. Since these factors are independent of the coupling between the SuperARV tagger and modifiee specification, we investigate their impact on a loosely integrated SCDG parser by comparing four models: (1) the basic loosely integrated model; (2) the basic model with crossing dependencies; (3) model 2 with distance and barrier information; (4) model 3 with SuperARVs augmented with additional modifiee lexical feature constraints. Each model uses a trigram SuperARV tagger to generate 40-best SuperARV sequences prior to modifiee specification. Table 2 shows the results for each of the four models including SuperARV tagging accuracy (%) and role value labeled precision and recall (%). Allowing crossing dependencies improves the overall parsing accuracy, but using distance information with verb barrier and punctuation heuristics produces an even greater improvement especially on the longer sentences. The accuracy is further improved by the additional modifiee lexical feature constraints added to the SuperARVs. Note that RLR is lower than RLP in these investigations possibly due to SuperARV tagging errors and the use of a tight stack pruning threshold.

Next, we evaluate the impact of increasing the context of the SuperARV tagger to a 4-gram while increasing the size of the N-best list passed from the tagger to the modifiee specification step of the parser. For this evaluation, we use model (4)

Table 2: Results on Section 23 of the WSJ Penn Treebank for four loosely-coupled model variations. The evaluation metrics, **RLR** and **RLP**, are our dependency-based role value labeled precision and recall. Note: Model (1) denotes the basic model, Model (2) denotes (1)+crossing dependencies, Model (3) denotes (2)+distance (punctuation) model, and Model (4) denotes (3)+modifi ee lexical features.

Models	$\leq$ **40 words (2,245 sentences)**				
	Tagging	**governor only**		**all roles**	
	Acc.	**RLP**	**RLR**	**RLP**	**RLR**
(1)	94.7	90.6	90.3	86.8	86.2
(2)	95.0	90.7	90.5	87.0	86.5
(3)	95.7	91.1	90.9	87.4	87.0
(4)	96.2	91.5	91.2	88.0	87.4
Models	$\leq$ **100 words (2,416 sentences)**				
	Tagging	**governor only**		**all roles**	
	Acc.	**RLP**	**RLR**	**RLP**	**RLR**
(1)	94.0	89.7	89.3	86.0	85.5
(2)	94.2	89.9	89.6	86.2	85.8
(3)	94.7	90.4	90.2	86.8	86.3
(4)	95.4	90.9	90.5	87.5	86.8

Table 3: Results on Section 23 of the WSJ Penn Treebank comparing models that utilize different SuperARV taggers and N-best sizes with the tightly coupled implementation. Note L denotes **Loose coupling** and T denotes **Tight coupling**. Also (a) denotes trigram, 40-best; (b) denotes trigram, 100-best; (c) denotes 4-gram, 40-best; (d) denotes 4-gram, 100-best.

Models		$\leq$ **40 words (2,245 sentences)**				
		Tagging	**governor only**		**all roles**	
		Acc.	**RLP**	**RLR**	**RLP**	**RLR**
L	(a)	96.2	91.5	91.2	88.0	87.4
	(b)	96.7	91.9	91.5	88.3	87.7
	(c)	96.9	92.2	91.7	88.6	88.1
	(d)	97.2	92.4	92.3	89.1	88.6
T		97.4	93.2	92.9	89.8	89.2
Models		$\leq$ **100 words (2,416 sentences)**				
		Tagging	**governor only**		**all roles**	
		Acc.	**RLP**	**RLR**	**RLP**	**RLR**
L	(a)	95.4	90.9	90.5	87.5	86.8
	(b)	95.8	91.3	90.8	87.7	87.0
	(c)	96.0	91.7	91.2	88.0	87.4
	(d)	96.3	91.8	91.5	88.5	87.8
T		96.6	92.6	92.2	89.1	88.5

from Table 2, the most accurate model so far. We also evaluate whether a tight integration of left-to-right SuperARV tagging and modifiee specification produces a greater parsing accuracy than the best loosely coupled counterpart. Table 3 shows the SuperARV tagging accuracy (%) and role value labeled precision and recall (%) for each model. Consistent with our intuition, a stronger SuperARV tagger and a larger search space of SuperARV sequences produces greater parse accuracy. However, tightly integrating SuperARV prediction with modifiee specification achieves the greatest overall accuracy. Note that SuperARV tagging accuracy and parse accuracy improve in tandem, as can be seen in Tables 2 and 3. These results are consistent with the observations of (Collins, 1999) and (Eisner, 1996). It is important to note that each of the factors contributing to improved parse accuracy in these two experiments also improved the word prediction capability of the corresponding parser-based LM (Wang and Harper, 2003).

5.2 Comparing to Other Parsers

Charniak's state-of-the-art PCFG parser (Charniak, 2000) has achieved the highest PARSEVAL LP/LR when compared to Collins' Model 2 and Model 3 (Collins, 1999), Roark's (Roark, 2001), Ratnaparkhi's (Ratnaparkhi, 1999), and Xu & Chelba's (Xu et al., 2002) parsers. Hence, we will compare our best loosely integrated and tightly integrated SCDG parsers to Charniak's parser. Additionally, we will compare with Collins' Model 2 since it makes the complement/adjunct distinction and Model 3 since it handles wh-movement (Collins, 1999). Charniak's parser does not explicitly model these phenomena.

Among the statistical CFG parsers to be compared, only Collins' Model 3 produces trees with information about wh-movement. Since the transformer uses empty node information to transform the CFG parse trees to CDG parses, the accuracy of Charniak's parser and Collins' Model 2 may be slightly reduced for sentences with empty nodes. Hence, we compare results on two test sets: one that omits all sentences with traces and one that does not. As can be seen in Table 4, our tightly coupled parser consistently produces an accuracy that equals or exceeds the accuracies of the other parsers, with one exception (Collins' Model 3), regardless of whether the test set contains sentences with traces.

Using our evaluation metrics, Collins' Model 3 achieves a better precision/recall than Model 2 and Charniak's parser. Since trace information is used by the CFG-to-CDG transformer to generate certain lexical features (Wang, 2003), the output from Model 3 is likely to be mapped to more accurate CDG parses. Although Charniak's maximum-entropy inspired parser achieved the highest PARSEVAL results, Collins' Model 3 is more accurate using our dependency metric, possibly because it makes the complement/adjunct distinction and models wh-movement. Since the statistical

Table 4: Evaluation of five models on Section 23 sentences with and without traces: **L** denotes the best loosely coupled CDG parser and **T** the tightly coupled CDG parser.

Models	**≤ 40 words (2,245 sentences)**							
	Without TRACE (1,903 sentences)				**All (2,245 sentences)**			
	governor only		**all roles**		**governor only**		**all roles**	
	RLP	**RLR**	**RLP**	**RLR**	**RLP**	**RLR**	**RLP**	**RLR**
L	92.4	92.4	89.5	88.7	92.4	92.3	89.1	88.6
T	93.2	92.9	89.9	89.3	93.2	92.9	89.8	89.2
Charniak (Charniak, 2000)	92.6	92.5	89.4	88.9	92.5	92.3	88.9	88.7
Collins, Model 2 (Collins, 1999)	92.5	92.3	89.1	88.5	92.2	92.1	89.0	88.5
Collins, Model 3 (Collins, 1999)	92.8	92.7	89.9	89.4	92.7	92.4	89.3	89.1
Models	**≤ 100 words (2,416 sentences)**							
	Without TRACE (1,979 sentences)				**All (2,416 sentences)**			
	governor only		**all roles**		**governor only**		**all roles**	
	RLP	**RLR**	**RLP**	**RLR**	**RLP**	**RLR**	**RLP**	**RLR**
L	91.9	91.6	88.8	88.1	91.8	91.5	88.5	87.8
T	92.7	92.3	89.4	88.7	92.6	92.2	89.1	88.5
Charniak (Charniak, 2000)	92.0	91.8	88.8	88.2	91.9	91.6	88.4	87.9
Collins, Model 2 (Collins, 1999)	91.8	91.6	88.6	88.0	91.7	91.5	88.2	87.9
Collins, Model 3 (Collins, 1999)	92.2	92.1	89.4	88.8	92.1	91.9	88.8	88.5

CFG parsers may loose accuracy from the CFG-to-CDG transformation, similarly to Collins' experiment reported in (Hajic et al., 1998), we also transformed our CDG parses to Penn Treebank style CFG parse trees and scored them using PARSEVAL. On the WSJ PTB test set, Charniak's parser achieved 89.6% LR and 89.5% LP, Collins' Model 2 and 3 obtained 88.1% LR and 88.3% LP and 88.0% LR and 88.3% LP, while the tightly coupled CDG parser obtains 85.8% LR and 86.4% LP. It is important to remember that this score is impacted by two lossy conversions, one for training and one for testing.

We have conducted a non-parametric Monte Carlo test to determine the significance of the differences between the parsing accuracy results in Table 3 and Table 4. We found that the difference between the tightly and loosely coupled SCDG parsers is statistically significant, as well as the difference between the SCDG parser and Charniak's parser and Collins' Model 2. Although the difference between our parser and Collins' Model 3 is not statistically significant, our parser represents a first attempt to build a high quality SCDG parser, and there is still room for improvement, e.g., better handling of barriers (including punctuation) and employing more sophisticated search and pruning strategies.

This paper has presented a statistical implementation of a CDG parser, which is both generative and highly lexicalized. With a framework of tightly integrated, multiple knowledge sources, model distance, and synergistic dependencies, we have achieved a parsing accuracy comparable to the state-of-the-art statistical parsers trained on the Wall Street Journal Penn Treebank corpus. However, more work must be done to build a parser model capable of coping with speech disfluencies present in spontaneous speech. We also intend to investigate a hybrid parser that combines the generality of a CFG with the specificity of a CDG.

References

E. Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*.

E. Charniak. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of the First Annual Meeting of the North American Association for Computational Linguistics*.

E. Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of ACL'2001*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

J. M. Eisner. 1996. An empirical comparison of probability models for dependency grammar. Technical report, University of Pennsylvania, CIS Department, Philadelphia PA 19104-6389.

J. Hajic, E. Brill, M. Collins, B. Hladka, D. Jones, C. Kuo, L. Ramshaw, O. Schwartz, C. Tillmann, and D. Zeman. 1998. Core natural language processing technology applicable to multiple languages – Workshop '98. Technical report, Johns Hopkins Univ.

M. P. Harper and R. A. Helzerman. 1995. Extensions to constraint dependency parsing for spoken language processing. *Computer Speech and Language*.

M. P. Harper, W. Wang, and C. M. White. 2001. Approaches for learning constraint dependency grammar from corpora. In *Proceedings of the Grammar and Natural Language Processing Conference*, Montreal, Canada.

F. Jelinek. 1997. *Statistical Methods For Speech Recognition*. The MIT Press.

D. Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1420–1427.

D. M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283.

A. Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34:151–175.

B. Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

W. Wang and M. P. Harper. 2002. The SuperARV language model: Investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proceedings of Conference of Empirical Methods in Natural Language Processing*.

W. Wang and M. P. Harper. 2003. Language modeling using a statistical dependency grammar parser. In *Proceedings of International Workshop on Automatic Speech Recognition and Understanding*.

W. Wang, M. P. Harper, and A. Stolcke. 2003. The robustness of an almost-parsing language model given errorful training data. In *ICASSP 2003*.

W. Wang, A. Stolcke, and M. P. Harper. 2004. The use of a linguistically motivated language model in conversational speech recognition. In *ICASSP 2004*.

W. Wang. 2003. *Statistical Parsing and Language Modeling based on Constraint Dependency Grammar*. Ph.D. thesis, Purdue University.

P. Xu, C. Chelba, and F. Jelinek. 2002. A study on richer syntactic dependencies for structured language modeling. In *Proceedings of ACL 2002*.

S. J. Young, J. Odell, D. Ollason, V. Valtchev, and P. C. Woodland, 1997. *The HTK Book*. Entropic Cambridge Research Laboratory, Ltd.