W. Wang and M. P. Harper, "Language Modeling Using a Statistical Dependency Grammar Parser," submitted to the 2003 International Workshop on Automatic Speech Recognition and Understanding.

LANGUAGE MODELING USING A STATISTICAL DEPENDENCY GRAMMAR PARSER

Wen Wang[1,2], *Mary P. Harper*[1]

[1] Electrical and Computer Engineering, Purdue University
West Lafayette, IN 47907-1285
[2] Speech Technology and Research Laboratory, SRI International
Menlo Park, CA 94025
wwang@speech.sri.com, harper@ecn.purdue.edu

ABSTRACT

Constraint Dependency Grammar (CDG) uses constraints to determine a sentence's grammatical structure that is represented as assignments of dependency relations to functional variables associated with each word in the sentence. This paper presents the evaluation of a statistical CDG parser-based language model (LM). This LM, when used to rescore lattices from the Wall Street Journal continuous speech recognition task, obtains a significant reduction in word error rate (WER) compared to a CDG-based almost-parsing LM and obtains a WER comparable to or lower than several state-of-the-art parser-based LMs.

1. INTRODUCTION

Recent language modeling efforts have explored the use of hidden parse trees to improve LM performance [1, 2, 3, 4]. Chelba et al. [1] built a left-to-right parser-based LM that develops hidden hierarchical structures incrementally to model long distance dependencies. Xu and Chelba [4] revised this model by enlarging the context window and using non-terminal information of siblings. Charniak [3] developed a parser-based LM by enriching non-terminal expansions with headword information. Roark's probabilistic top-down parser-based LM [2] incrementally calculates generative conditional word probabilities similarly to left-corner parsers. All of these approaches share an underlying context-free grammar (CFG) based framework; the parser component of each LM builds constituents that are labeled with enriched non-terminals that often correspond to phrases in an utterance.

This research was supported in part by Purdue Research Foundation, National Science Foundation under Grant No. BCS-9980054, and DARPA under Contract No. MDA972-02-C-0038. These experiments were conducted by the first author while working at SRI International. Part of this work was carried out while the second author was on leave at National Science Foundation. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the view of DARPA or the National Science Foundation. The authors would like to thank researchers of Speech Technology and Research Laboratory of SRI International for their suggestions and support.

Much of our recent research is predicated on our belief that avoiding the use of non-terminals, and instead directly using word-to-word dependencies to represent a parse tree, can be advantageous for building high quality parser-based LMs. Hence, our work has focused on constructing LMs based on Constraint Dependency Grammar (CDG) [5], which represents syntactic structures using 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). Note that the assignment of role values to roles for each word defines the parse for the sentence. Consider the role value assigned to the governor role (denoted G) of *you*, `subj-2`. The label `subj` indicates the grammatical function of *you* when it is governed by its head in position `2`. Need roles are used to ensure that the grammatical requirements of a word are met (e.g., subcategorization). The word *you* does not have any need roles (denoted $N1$, $N2$, and $N3$) unlike the verb *did* that needs a subject and a base form verb (but since the word takes no other complements, the modifiee of the role value assigned to N3 is set equal to its own position).

Because CDGs represent parse information at the word level, there is no need to model non-terminals. CDG can be easily lexicalized at the word level, and this lexicalization is able to include not only lexical category, but also a rich set of lexical features to model subcategorization and wh-movement without a combinatorial explosion of the parametric space [5]. CDG can distinguish between adjuncts and complements due to the use of need roles [6]. CDG is more powerful than CFG and has the ability to model languages with crossing dependencies and free word ordering; hence, this research in English has a potential to generalize to a wide variety of languages.

In an earlier effort [5], we developed an almost-parsing LM based on CDG. The underlying hidden event of this

LM is a *SuperARV*, a linguistic structure that tightly integrates lexical features and syntactic constraints at the word level. A SuperARV, an abstraction of the joint assignment of all dependencies for a word, 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). The SuperARV LM reduced perplexity and word error rate significantly compared to word ngrams and part-of-speech-based LMs and obtained significant perplexity reduction compared to several parser-based LMs [5]. Additionally, it obtained WER rates slightly lower than (but not significantly different from) the state-of-the-art Chelba's structured LM [1].

A sentence tagged with SuperARVs is an almost-parse since all that remains is to specify the precise position of each modifiee. In this paper, we will investigate the quality of a LM that is based upon a statistical constraint dependency grammar (SCDG) parser, which combines SuperARV tagging with modifiee specification. We believe that this SCDG LM will have greater restrictiveness and predictive capability than a SuperARV LM. Section 2 outlines our SCDG parser-based LM. Section 3 describes the evaluation of the LM, compares it to other parser-based LMs, and discusses the results.

2. THE SCDG PARSER-BASED LM

Like other parser-based LMs, an SCDG parser LM estimates a string's probability by summing the probabilities of all CDG parses for the string produced by the parser. Given that the set of partial CDG parses for a sentence prefix w_1^{i-1} is D_1^{i-1}, the probability $P(w_i|w_1^{i-1})$ is calculated as:

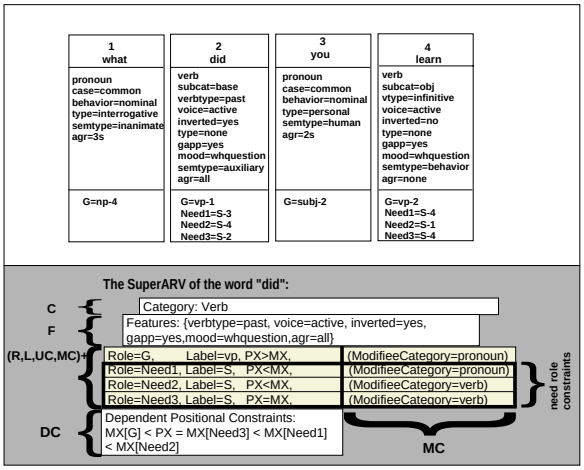

Fig. 1. An example of a CDG parse and the SuperARV of the word *did* in the sentence *what did you learn*. PX and MX([R]) represent the position of a word and its modifiee (for role R), respectively.

$$P_{\text{SCDG}}(w_i|w_1^{i-1}) = \frac{\sum_{d \in D_1^i} P(d)}{\sum_{d' \in D_1^{i-1}} P(d')} \quad (1)$$

where $P(d)$ is the probability of a partial CDG parse d for the sentence prefix w_1^i in question. In section 2.1, we describe the implementation of the underlying SCDG parser, discussing, in particular, how to calculate $P(d)$ in Equation (1). In section 2.2, we discuss parser model enhancements that are possible to produce a more powerful SCDG LM.

2.1. The Basic Parsing Algorithm

Our SCDG parser is a probabilistic generative model. For all sentences *s* and all parses *T*, the parser assigns a probability $Pr(s,T) = P(T)$ for all T with yield s. For s, the parser returns the parse T that maximizes its probability, as follows:

$$\begin{aligned} argmax_T Pr(T|s) &= argmax_T Pr(s,T) \\ &= argmax_T Pr(T) \end{aligned}$$

Our statistical parser 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 shown in Figure 2, and the symbols used in the algorithm are described 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 [7] 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 their 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[1]. The values of syn 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 algorithm is implemented as a simple best-first 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 [9]. To simplify presentation, we describe the smoothing procedure for the loosely coupled case, where the $+$ prefix represents items related to right dependents and the $-$ prefix, left dependents:

$$\begin{aligned}
& Pr(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \mathcal{H}) \\
= & \sum_{i=1}^{c} \eta_i \cdot Pr(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \delta_i) \\
+ & \lambda_1 \cdot Pr(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \mathcal{H}_1) \\
+ & \lambda_2 \cdot Pr(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \mathcal{H}_2) \\
+ & \lambda_3 \cdot Pr(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \mathcal{H}_3) \\
+ & \lambda_4 \cdot Pr(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \mathcal{H}_4) \\
+ & \lambda_5 \cdot Pr(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \mathcal{H}_5) \\
+ & \lambda_6 \cdot Pr(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \mathcal{H}_6)
\end{aligned}$$

where,

$$\begin{aligned}
\mathcal{H} &= \langle w, s\rangle_k, \langle w, s\rangle_{dep(k,\pm(c+1))}, \langle w, s\rangle_{dep(k,\pm1)}^{dep(k,\pm c)} \\
\delta_i &= \langle w, s\rangle_k, \langle w, s\rangle_{dep(k,\pm(c+1))}, \langle w, s\rangle_{\pm1}^{\pm i} \\
\mathcal{H}_1 &= \langle w, s\rangle_k, w_{dep(k,\pm(c+1))}, s_{dep(k,\pm1)}^{dep(k,\pm c)} \\
\mathcal{H}_2 &= \langle w, s\rangle_k, s_{dep(k,\pm(c+1))}, s_{dep(k,\pm1)}^{dep(k,\pm c)} \\
\mathcal{H}_3 &= s_k, \langle w, s\rangle_{dep(k,\pm(c+1))}, s_{dep(k,\pm1)}^{dep(k,\pm c)} \\
\mathcal{H}_4 &= s_k, s_{dep(k,\pm(c+1))}, s_{dep(k,\pm1)}^{dep(k,\pm c)} \\
\mathcal{H}_5 &= s_k, s_{dep(k,\pm(c+1))}, POS_{dep(k,\pm1)}^{dep(k,\pm c)} \\
\mathcal{H}_6 &= POS_k, POS_{dep(k,\pm(c+1))}, POS_{dep(k,\pm1)}^{dep(k,\pm c)}
\end{aligned}$$

Note POS_k denotes the part-of-speech of word w_k. The values of η_i's and λ's are determined using the EM algorithm described by Jelinek [9] on a held-out data set.

2.2. Additions to the Basic Model

Some additional features that improve SCDG parsing accuracy are added to the basic model because of their potential to improve the word prediction capability of the SCDG LM. These additions appear in this subsection, and their efficacy is evaluated in Section 3.

1. **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 particular.
2. **Distance and barriers between dependents:** Because distance between two dependent words (e.g., the distance value 0 means the two words are adjacent) 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 as: $[0]$, $[1]$, $[2, 4]$, $[5, 9]$, $[10, \infty)$. A discrete random variable with five possible ranges is used to model distance. Collins [11] observed that 94% of dependencies do not cross a verb in the Wall Street Journal Penn Treebank training set. This suggests that verbs act as barriers that impact modifiee links. Hence,

[1] This is a much simpler mechanism than the *trigger* approach used by Rosenfeld [8] to model synergistic features.

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, repectively
$dep(k,-N(\mathcal{L}(s_k)))$, $dep(k,N(\mathcal{L}(s_k)))$	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

a Boolean random variable that represents whether there is a verb between the dependencies is added to conditions of the probability estimations.

3. **Modifiee lexical features:** The SuperARV structure employed in the CDG-based almost-parsing LM [5] uses only lexical categories of modifiees as modifiee constraints. In our previous work [12], we found that modifiee lexical features play an important role in strengthening selectivity of a CDG. However, for many dependencies, only a subset of lexical features are relevant to the dependency relation. Consequently, we investigate the efficacy of adding relevant lexical features to modifiee constraints for a SuperARV structure.

3. EVALUATIONS AND DISCUSSIONS

3.1. Experimental Setup

The 20k open vocabulary DARPA Hub1 WSJ CSR is used to evaluate our SCDG parser-based LMs. Roark [2], Chelba [1], and Xu & Chelba [4] all evaluated their parser-based LMs on this task. The LM training data for this task is composed of the 1987-1989 files containing 37,243,300 words. Since this is a speech corpus, this material contains no punctuation or case information. All words outside the provided vocabulary are mapped to $\langle UNK\rangle$. We evaluate all LMs on the 1993 20k open vocabulary DARPA WSJ CSR evaluation set (denoted 93-20K), which consists of 213 utterances and 3,446 words. For consistency, we use the same lattices as those used by Roark [2], Chelba [1], and Xu & Chelba [4]. Note that, for the parser-based LMs, we split the contractions in the word lattices so that they can be processed properly. We will evaluate our SCDG parser-based LM on this set of lattices using lattice rescoring.

Our SCDG parser LM was trained using CDG parses for the complete WSJ CSR LM training data, i.e., we utilize all of the 37+ million word data set. To obtain these parses, we used an automatic transformer (as in [5]) to transform the CFG parses into CDG parses. The CFG parses were obtained from a combination of the WSJ Penn Treebank [13] and BLLIP [14], which together cover the entire LM training set. BLLIP was generated using Charniak's maximum entropy inspired parser [15] trained on the WSJ Penn Treebank. Since WSJ Penn Treebank parses are probably more accurate than those contained in BLLIP, the parse trees for training our LM were generated following the policy that for each training set sentence, if the parse can be found in the WSJ PTB, it becomes the parse tree for the training sentence; otherwise, we use the corresponding tree from the BLLIP treebank [14].

We compare several variations of our SCDG parser LM to other LMs listed below. All are trained on the complete 37+ million word LM training data using the 20K open vocabulary, unless mentioned otherwise.

- The baseline trigram provided by LDC for the task.
- The part-of-speech (POS) LM described in [5].
- The best SuperARV LM described in [5].
- Chelba's parser-based LM [1], which was trained on a 20-million-word subset of the WSJ CSR LM training data due to time and space constraints on the model.
- Chelba's parser-based LM which we retrained using more optimization and the entire training set [5].
- Xu and Chelba's revised parser-based LM [4], which was trained on a 20-million-word subset of the WSJ CSR LM training data due to time and space constraints on the model.

Note that we do not compare to Roark's model [2] since his model was trained on the much smaller Wall Street Journal Penn Treebank corpus. Note that Xu & Chelba [4] rescored 50-best lists generated from Chelba's lattices [1]; whereas, Chelba [1] performed lattice rescoring.

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.

Fig. 2. The basic loosely coupled parsing algorithm.

3.2. Results

Table 2 shows the word error rate (WER) and sentence accuracy (SAC) after applying each LM to rescore the lattices (or N-best lists). Note **Lattice WER/SAC** shows the best possible accuracy for the set of lattices used given perfect knowledge. As can be seen from the table, all of the SCDG parser LMs outperform the trigram, POS, and SuperARV LMs. For the loosely coupled SCDG LMs, the model enrichment improves the LM performance. Also, the tightly coupled SCDG parser-based LM outperforms the best loosely coupled SCDG parser-based LMs, which verifies our hypothesis that this tight integration should benefit both tagging and modifiee determination. Also, all of the loosely and tightly coupled SCDG LMs obtain a WER reduction and SAC increase compared to Chelba's original LM [1]. Furthermore, after lattice rescoring, the tightly coupled LM performs slightly better than Xu and Chelba's revised structured LM [4].

It should be noted that Xu and Chelba [4] trained their LM on a subset of the WSJ CSR training data and rescored 50-best lists derived from the same set of lattices. Consequently, to more directly compare our tightly coupled SCDG parser-based LM to this state-of-the-art LM, we retrained it on the smaller set of training data used by [4] and applied the resulting LM to rescore 50-best lists. In this case, we obtained a WER of 12.5%, a higher WER than 12.3% reported by Xu and Chelba. However, when the tightly coupled SCDG parser LM was trained on the complete data set, we obtained a comparable WER of 12.28% given 50-best rescoring.

There are two important characteristics of grammars [16] that can affect the quality of a parser-based LM and clearly have a tradeoff: **generality** and **selectivity**. For example, a probabilistic CFG without headword enrichment would be quite general but would not have the selectivity necessary for constructing a high quality LM. In prior work on CDG induction [12], we found that CDGs tend to have a greater selectivity and lower generality than CFGs learned from the same corpus. The LM results described above are consistent with our findings concerning differences in inductive bias between CFGs and CDGs. Although it is possible to relax grammar constraints to increase generality, we hypothesize that a selective SCDG parser LM trained on more data should perform better, as shown in our results.

Models		WER(SAC) (%)	Interp. Weight
3gram		13.72(36.18)	N/A
POS		13.51(37.96)	1.0
SuperARV		12.83(43.86)	1.0
Loose SCDG LM	**(1): basic model**	12.79(44.25)	0.8
	(2): (1)+crossing dependencies	12.71(45.86)	0.8
	(3): (2)+distance and barrier model	12.59(48.33)	0.8
	(4): (3)+modifiee lexical features	12.44(49.02)	0.8
Tight SCDG LM using (4)		12.18(51.66)	0.9
Chelba (re-run)		12.89(41.66)	0.6
Chelba (2000)		13.0(–)	0.6
Xu and Chelba (2002)		12.3(–)	0.6
lattice accuracy		3.41 (68.86)	N/A

Table 2. Comparing WER and SAC (%) after rescoring lattices or N-best lists using each LM on the DARPA WSJ CSR 1993 20K open vocabulary evaluation test set. The lattice WER and SAC which define the best accuracy possible for a set of lattices given perfect knowledge is also provided.

Table 2 shows the interpolation weight λ for combining each corresponding model with the baseline trigram (optimized on the same development set). As discussed in [5], the POS and almost-parsing SuperARV LMs are not improved by interpolation because they already contain word co-occurrence knowledge in their models. By contrast, the parser-based LMs [1, 4] obtain a decrease in WER when they are interpolated with word n-gram LMs. Because of their focus on modeling syntactic knowledge with non-terminals, they may not capture all of the local word co-occurrence information available to the trigram. Our SCDG parser-based LM tightly integrates word and parse structure information; however, all of the SCDG parser-based LMs, whether loosely or tightly coupled with SuperARV tagging, obtain an improved performance given interpolation with the trigram. The lower interpolation weight of the loosely coupled SCDG parser-based LM indicates that it receives a greater level of compensation from the word n-gram than its tightly coupled counterpart. The SCDG parser-based LMs may be effectively interpolated with the trigram due to the distortion of probability mass due to pruning.

This paper has presented a statistical Constraint Dependency Grammar parser-based LM that tightly integrates multiple knowledge sources such as word identity, syntactic constraints, and lexical features. The model also utilized long-distance dependency information and subcategorization information to make word predictions. When evaluated on the WSJ CSR task, the model outperformed an almost-parsing statistical LM based on CDG and produced a recognition accuracy comparable to or exceeding state-of-the-art parser-based LMs.

4. REFERENCES

[1] C. Chelba, *Exploiting Syntactic Structure for Natural Language Modeling*, Ph.D. thesis, Johns Hopkins University, 2000.

[2] B. Roark, "Probabilistic top-down parsing and language modeling," *Computational Linguistics*, vol. 27, no. 2, pp. 249–276, 2001.

[3] E. Charniak, "Immediate-head parsing for language models," in *Proceedings of ACL*, 2001.

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

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

[6] M. P. Harper and R. A. Helzerman, "Extensions to constraint dependency parsing for spoken language processing," *Computer Speech and Language*, vol. 9, pp. 187–234, 1995.

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

[8] R. Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?," *Proceedings of the IEEE*, vol. 88, pp. 1270–1278, August 2000.

[9] F. Jelinek, *Statistical Methods For Speech Recognition*, The MIT Press, 1997.

[10] M. J. Collins, "A new statistical parser based on bigram lexical dependencies," in *Proceedings of ACL*, 1996, pp. 184–191.

[11] M. P. Harper and W. Wang, "Approaches for learning constraint dependency grammar from corpora," in *Proceedings of the Grammar and Natural Language Processing Conference*, Montreal, Canada, 2001.

[12] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[13] E. Charniak, D. Blaheta, N. Ge, K. Hall, and M. Johnson, "BLLIP WSJ corpus," CD-ROM, 2000, Linguistics Data Consortium.

[14] E. Charniak, "A maximum-entropy-inspired parser," in *Proceedings of NAACL*, 2000.

[15] J. Allen, *Natural Language Understanding*, Benjamin/Cummings Publishing Company, Redwood City, 1995.