The Effectiveness of Corpus-Induced Dependency Grammars for Post-processing Speech*

M. P. Harper, C. M. White, W. Wang, M. T. Johnson, and R. A. Helzerman
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285
{harper,robot,wang28,mjohnson,helz}@ecn.purdue.edu

Abstract

This paper investigates the impact of Constraint Dependency Grammars (CDG) on the accuracy of an integrated speech recognition and CDG parsing system. We compare a conventional CDG with CDGs that are induced from annotated sentences and template-expanded sentences. The grammars are evaluated on parsing speed, precision/coverage, and improvement of word and sentence accuracy of the integrated system. Sentence-derived CDGs significantly improve recognition accuracy over the conventional CDG but are less general. Expanding the sentences with templates provides us with a mechanism for increasing the coverage of the grammar with only minor reductions in recognition accuracy.

1 Background

The question of when and how to integrate language models with speech recognition systems is gaining in importance as recognition tasks investigated by the speech community become increasingly more challenging and as speech recognizers are used in human/computer interfaces and dialog systems (Block, 1997; Pieraccini and Levin, 1992; Schmid, 1994; Wright et al., 1994; Zue et al., 1991). Many systems tightly integrate N-gram stochastic language models, with a power limited to a regular grammar, into the recognizer (Jeanrenaud et al., 1995; Ney et al., 1994; Placeway et al., 1993) to build more accurate speech recognizers. However, in order to act based on the spoken interaction with the user, the speech signal must be mapped to an internal representation. Obtaining a syntactic representation for the spoken utterance has a high degree of utility for mapping to a semantic representation. Without a structural analysis of an input, it is difficult to guarantee the correctness of the mapping from a sentence to its interpretation (e.g., mathematical expressions to internal calculations). We believe that significant additional improvement in accuracy can be gained in specific domains by using a more complex language model that combines syntactic, semantic, and domain knowledge.

A language processing module that is more powerful than a regular grammar can be loosely, moderately, or tightly integrated with the spoken language system, and there are advantages and disadvantages associated with each choice (Harper et al., 1994). To tightly integrate a language model with the power of a context-free grammar with the acoustic module requires that the power of the two modules be matched, making the integrated system fairly intractable and difficult to train. By separating the language model from the acoustic model, it becomes possible to use a more powerful language model without increasing computational costs or the amount of acoustic training data required by the recognizer. Furthermore, a loosely-integrated language model can be developed independently of the speech recognition component, which is clearly an advantage. Decoupling the acoustic and language models also adds flexibility: a wide variety of language models can be tried with a single acoustic model. Systems that utilize a language model that operates as a post-processor to a speech recognizer include (Block, 1997; Seneff, 1992; Zue et al., 1991).

The goal of this research is to construct and experimentally evaluate a prototype of a spoken language system that loosely integrates a speech recognition component with an NLP component that uses syntactic, semantic, and domain-specific knowledge to more accurately select the sentence uttered by a speaker. First we describe the system we have built. Then we describe the mechanism used to rapidly develop a domain-specific grammar that improves accuracy of our speech recognizer.

2 Our System

We have developed the prototype spoken language system depicted in Figure 1 that integrates a speech recognition component based on HMMs with a powerful grammar model based on Constraint Dependency Grammar (CDG). The speech recognizer is implemented as a multiple-mixture triphone HMM with a simple integrated word co-occurrence gram-

* This research was supported by grants from Intel, Purdue Research Foundation, and National Science Foundation IRI 97-04358, CDA 96-17388, and #9980054-BCS.

mar (Ent, 1997; Young et al., 1997). Mel-scale cepstral coefficients, energy, and each of their their first and second order differences are used as the underlying feature vector for each speech frame. Model training is done using standard Baum-Welch Maximum Likelihood parameter re-estimation on diagonal covariance Gaussian Mixture Model (GMM) feature distributions. The speech recognizer employs a token-passing version of the Viterbi algorithm (Young et al., 1989) and pruning settings to produce a pruned recognition lattice. This pruned lattice contains the most likely alternative sentences that account for the sounds present in an utterance as well as their probabilities. Without any loss of information, this lattice is then compressed into a word graph (Harper et al., 1999b; Johnson and Harper, 1999), which acts as the interface between the recognizer and the CDG parser. The word graph algorithm begins with the recognition lattice and eliminates identical subgraphs by iteratively combining word nodes that have exactly the same preceding or following nodes (as well as edge probabilities), pushing excess probability to adjacent nodes whenever possible. The resulting word graph represents all possible word-level paths without eliminating or adding any paths or modifying their probabilities. Word graphs increase the bandwidth of useful acoustic information passed from the HMM to the CDG parser compared to most current speech recognition systems.

The CDG parser parses the word graph to identify the best sentence consistent with both the acoustics of the utterance and its own additional knowledge. The loose coupling of the parser with the HMM allows us to construct a more powerful combined system without increasing the amount of training data for the HMM or the computational complexity of either of the component modules. Our NLP component is implemented using a CDG parser (Harper and Helzerman, 1995; Maruyama, 1990a; Maruyama, 1990b) because of its power and flexibility, in particular:

- It supports the use of syntactic, semantic, and domain-specific knowledge in a uniform framework.
- Our CDG parser supports efficient simultaneous parsing of alternative sentence hypotheses in a word graph (Harper and Helzerman, 1995; Helzerman and Harper, 1996).
- Because CDG is a dependency grammar, it can better model free-order languages. Hence, CDG can be used in processing a wider variety of human languages than other grammar paradigms.
- It is capable of representing and using context-dependent information unlike traditional grammar approaches, thus providing a finer degree of control over the syntactic analysis of a sentence.
- A CDG can be extracted directly from sentences annotated with dependency information (i.e., feature and syntactic relationships).

We hypothesize that the accuracy of the combined HMM/CDG system should benefit from the ability to create a grammar that covers the domain as precisely as possible and that does not consider sentences that would not make sense given the domain. A corpus-based grammar is likely to have this degree of control. In the next section we describe how we construct a CDG from corpora.

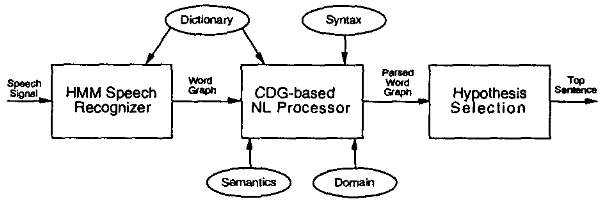


Figure 1: Block diagram of the loosely-coupled spoken language system.

3 Learning CDG Rules

In this section, we introduce CDG and then describe how CDG constraints can be learned from sentences annotated with grammatical information.

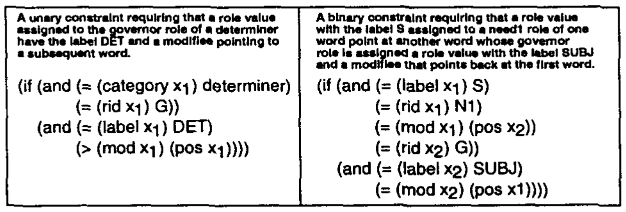
3.1 Introduction to CDG

Constraint Dependency Grammar (CDG), first introduced by Maruyama (Maruyama, 1990a; Maruyama, 1990b), uses constraints to determine the grammatical dependencies for a sentence. The parsing algorithm is framed as a constraint satisfaction problem: the rules are the constraints and the solutions are the parses. A CDG is defined as a five-tuple, $\langle \Sigma, R, L, C, T \rangle$, where $\Sigma = \{\sigma_1, \ldots, \sigma_c\}$ is a finite set of lexical categories (e.g., determiner), $R = \{r_1, \ldots, r_p\}$ is a finite set of uniquely named roles or role ids (e.g., governor, need1, need2), $L = \{l_1, \ldots, l_q\}$ is a finite set of labels (e.g., subject), C is a constraint formula, and T is a table that specifies allowable category-role-label combinations. A *sentence* $s = w_1 w_2 w_3 \ldots w_n$ has length n and is an element of Σ^*. For each word $w_i \in \Sigma$ of a sentence s, there are up to p different roles (with most words needing only one or two (Harper et al., 1999a)), yielding a maximum of $n * p$ roles for the entire sentence. A *role* is a variable that is assigned a *role value*, an element of the set $L \times \{1, 2, \ldots, n\}$. Role values are denoted as *l-m*, where $l \in L$ and $m \in \{1, 2, \ldots, n\}$ is called the *modifiee*. Maruyama originally used a modifiee of NIL to indicate that a role value does not require a modifiee, but it is more parsimonious to indicate that there is no dependent by setting the modifiee to the position of its word.

Role values are assigned to roles to record the syntactic dependencies between words in the sentence.

The governor role is assigned role values such that the modifiee of the word indicates the position of the word's governor or head (e.g., DET-3, when assigned to the governor role of a determiner, indicates its function and the position of its head). Every word in a sentence has a governor role. Need roles are used to ensure the requirements of a word are met. For example, an object is required by a verb that subcategorizes for one, unless it has passive voice. The required object is accounted for by requiring the verb's need role to be assigned a role value with a modifiee that points at the object. Words can have more than one need role, depending on the lexical category of the word. The table T indicates the roles that a word with a particular lexical category must support.

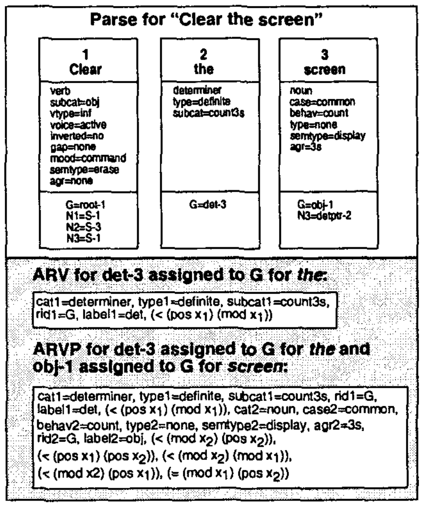
A sentence s is said to be **generated** by the grammar G if there exists an assignment $\mathbf{A}$ that maps a role value to each of the roles for s such that C is satisfied. There may be more than one assignment of role values to the roles of a sentence that satisfies C, in which case there is ambiguity. C is a first-order predicate calculus formula over all roles that requires that an assignment of role values to roles be consistent with the formula; those role values inconsistent with C can be eliminated. A subformula P_i of C is a predicate involving $=$, $<$, or $>$, or predicates joined by the logical connectives **and**, **or**, **if**, or **not**. A subformula is a *unary constraint* if it contains only a single variable (by convention, we use x_1) and a *binary constraint* if it contains two variables (by convention x_1 and x_2). An example of a unary and binary constraint appears in Figure 2. A CDG has an *arity* parameter a, which indicates the maximum number of variables in the subformulas of C, and a *degree* parameter d, which is the number of roles in the grammar. An arity of two suffices to represent a grammar at least as powerful as a context-free grammar (Maruyama, 1990a; Maruyama, 1990b). In (Harper et al., 1999a), we developed a way to write constraints concerning the category and feature values of a modifiee of a role value (or role value pair). These constraints loosely capture binary constraint information in unary constraints (or beyond binary for binary constraints) and results in more efficient parsing.



A unary constraint requiring that a role value assigned to the governor role of a determiner have the label DET and a modifiee pointing to a subsequent word.	A binary constraint requiring that a role value with the label S assigned to a need1 role of one word point at another word whose governor role is assigned a role value with the label SUBJ and a modifiee that points back at the first word.
(if (and (= (category x_1) determiner) (= (rid x_1) G)) (and (= (label x_1) DET) (> (mod x_1) (pos x_1))))	(if (and (= (label x_1) S) (= (rid x_1) N1) (= (mod x_1) (pos x_2)) (= (rid x_2) G)) (and (= (label x_2) SUBJ) (= (mod x_2) (pos x1))))

Figure 2: A Unary and binary constraint for CDG.

The white box in Figure 3 depicts a parse for the sentence *Clear the screen* from the Resource Management corpus (Price et al., 1988) (the ARV and ARVP in the gray box will be discussed later), which is a corpus we will use to evaluate our speech processing system. We have constructed a conventional CDG with around 1,500 unary and binary constraints (i.e., its arity is 2) that were designed to parse the sentences in the corpus. This CDG covers a wide variety of grammar constructs (including conjunctions and wh-movement) and has a fairly rich semantics. It uses 16 lexical categories, 4 roles (so its degree is 4), 24 labels, and 13 lexical feature types (**subcat**, **agr**, **case**, **vtype** (e.g., progressive), **mood**, **gap**, **inverted**, **voice**, **behavior** (e.g., mass), **type** (e.g., interrogative, relative), **semtype**, **takesdet**, and **conjtype**). The parse in Figure 3 is an assignment of role values to roles that is consistent with the unary and binary constraints. A role value, when assigned to a role, has access to not only the label and modifiee of its role value, but also the role name of the role to which it is assigned, information specific to the word (i.e., the word's position in the sentence, its lexical category, and feature values for each feature), and information about the lexical class and feature values of its modifiee. Our unary and binary constraints use this information to eliminate ungrammatical assignments.



Parse for "Clear the screen"

1 Clear	2 the	3 screen

verb
subcat=obj
vtype=inf
voice=active
inverted=no
gap=none
mood=command
semtype=erase
agr=none
	determiner
	type=definite
	subcat=count3s
		noun
		case=common
		behav=count
		type=none
		semtype=display
		agr=3s

G=root-1
N1=S-1
N2=S-3
N3=S-1
	G=det-3	G=obj-1
		N3=detptr-2

ARV for det-3 assigned to G for *the*:

cat1=determiner, type1=definite, subcat1=count3s, rid1=G, label1=det, (< (pos x_1) (mod x_1))

ARVP for det-3 assigned to G for *the* and obj-1 assigned to G for *screen*:

cat1=determiner, type1=definite, subcat1=count3s, rid1=G, label1=det, (< (pos x_1) (mod x_1)), cat2=noun, case2=common, behav2=count, type2=none, semtype2=display, agr2=3s, rid2=G, label2=obj, (< (mod x_2) (pos x_2)), (< (pos x_1) (pos x_2)), (< (mod x_2) (mod x_1)), (< (mod x2) (pos x_1)), (= (mod x_1) (pos x_2))

Figure 3: A CDG parse (see white box) is represented by the assignment of role values to roles associated with a word with a specific lexical category and one feature value per feature. ARVs and ARVPs (see gray box) represent grammatical relations that can be extracted from a sentence's parse.

3.2 Learning CDG Constraints

The grammaticality of a sentence in a language defined by a CDG was originally determined by applying the constraints of the grammar to the possible role value assignments. If the set of all possible role values assigned to the roles of a sentence of length n is denoted $\mathcal{S}_1 = \Sigma \times R \times POS \times L \times MOD \times F_1 \times \ldots \times F_k$, where k is the number of feature types, F_i represents the set of feature values for that type, $POS = \{1, 2, \ldots, n\}$ is the set of possible positions, $MOD = \{1, 2, \ldots, n\}$ is the set of possible modifiees, and n is sentence length (which can be any arbitrary natural number), then unary constraints partition $\mathcal{S}_1$ into grammatical and ungrammatical role values. Similarly, binary constraints partition the set $\mathcal{S}_2 = \mathcal{S}_1 \times \mathcal{S}_1 = \mathcal{S}_1^2$ into compatible and incompatible pairs. Building upon this concept of role value partitioning, it is possible to construct another way of representing unary and binary constraints because CDG constraints do not need to reference the exact position of a word or a modifiee in the sentence to parse sentences (Harper and Helzerman, 1995; Maruyama, 1990a; Maruyama, 1990b; Menzel, 1994; Menzel, 1995).

To represent the relative, rather than the absolute, position information for the role values in a grammatical sentence, it is only necessary to represent the positional relations between the modifiees and the positions of the role values. To support an arity of 2, these relations involve either equality or less-than relations over the modifiees and positions of role values assigned to the roles x_1 and x_2. Since unary constraints operate over role values assigned to a single role, the only relative position relations that can be tested are between the role value's position (denoted as $\mathrm{P}x_1$) and its modifiee (denoted as $\mathrm{M}x_1$); one and only one of the following three relations must be true: ($\mathrm{P}x_1 < \mathrm{M}x_1$), ($\mathrm{M}x_1 < \mathrm{P}x_1$), or ($\mathrm{P}x_1 = \mathrm{M}x_1$). Since binary constraints operate over role values assigned to pairs of roles, x_1 and x_2, the only possible relative position relations that can be tested are between $\mathrm{P}x_1$ and $\mathrm{M}x_1$, $\mathrm{P}x_2$ and $\mathrm{M}x_2$, $\mathrm{P}x_1$ and $\mathrm{M}x_2$, $\mathrm{P}x_2$ and $\mathrm{M}x_1$, $\mathrm{P}x_1$ and $\mathrm{P}x_2$, $\mathrm{M}x_1$ and $\mathrm{M}x_2$. Note that each of the six has three positional relations (as in the case of unary relations on $\mathrm{P}x_1$ and $\mathrm{M}x_1$) such that one and only one of them is simultaneously true.

The unary and binary positional relations provide the necessary mechanism to develop an alternative view of the unary and binary constraints. First, we develop the concept of an *abstract role value* (ARV), which is a finite characterization of all possible role values using relative, rather than absolute, position relations. Formally, an ARV for a particular grammar $G = \langle \Sigma, R, L, C, T, F_1, \ldots, F_k \rangle$ is an element of the set: $\mathcal{A}_1 = \Sigma \times R \times L \times F_1 \times \ldots \times F_k \times \mathrm{UC}$, where UC encodes the three possible positional relations between Px_1 and Mx_1. The gray box of Figure 3 shows an example of an ARV obtained from the parsed sentence. Note that $\mathcal{A}_1$ is a finite set representing the space of all possible ARVs for the grammar[1]; hence, the set provides an alternative characterization of the unary constraints for the grammar, which can be partitioned into positive (grammatical) and negative (ungrammatical) ARVs. During parsing, if a role value does not match one of the elements in the positive ARV space, then it should be disallowed. Positive ARVs can be obtained directly from the parses of sentences: for each role value in a parse for a sentence, simply extract its category, feature, role, and label information, and then determine the positional relation that holds between the role value's position and modifiee.

Similarly the set of legal *abstract role value pairs* (ARVPs), $\mathcal{A}_2 = \Sigma \times R \times L \times F_1 \times \ldots \times F_k \times \Sigma \times R \times L \times F_1 \times \ldots \times F_k \times \mathrm{BC}$, where BC encodes the positional relations among Px_1, Mx_1, Px_2, and Mx_2, provides an alternative definition for the binary constraints[2]. The gray box of Figure 3 shows an example of an ARVP obtained from the parsed sentence. Positive ARVPs can be obtained directly from the parses of sentences. For each pair of role values assigned to different roles, simply extract their category, feature, role, and label information, and then determine the positional relations that hold between the positions and modifiees.

An enumeration of the positive ARV/ARVPs can be used to represent the CDG constraints, C, and ARV/ARVPs are PAC-learnable from positive examples, as can be shown using the techniques of (Natarajan, 1989; Valiant, 1984). ARV/ARVP constraints can be enforced by using a fast table lookup method to see if a role value (or role value pair) is allowed (rather than propagating thousands of constraints), thus speeding up the parser.

4 Evaluation Using the Naval Resource Management Domain

An experiment was conducted to determine the plausibility and the benefits of extracting CDG constraints from a domain-specific corpus of sentences. For our speech application, the ideal CDG should be general enough to cover sentences similar to those that appear in the corpus while being restrictive enough to eliminate sentences that are implausible given the observed sentences. Hence, we investigate whether a grammar extracted from annotated sentences in a corpus achieves this precision of coverage. We also examine whether a learned grammar has the ability to filter out incorrect sentence hypotheses produced by the HMM component of our system in Figure 1. To investigate these issues, we have performed an experiment using the standard

[1] $\mathcal{A}_1$ can also include information about the possible lexical categories and feature values of the modifiee of x_1.

[2] $\mathcal{A}_2$ can also include information about the possible lexical categories and feature values of the modifiees of x_1 and x_2.

Resource Management (RM) (Price et al., 1988) and Extended Resource Management (RM2) ((DARPA), 1990) corpora. These mid-size speech corpora have a vocabulary of 991 words and contain utterances of sentences derived from sentence templates based on interviews with naval personnel familiar with naval resource management tasks. They were chosen for several reasons: they are two existing speech corpora from the same domain; their manageable sizes make them a good platform for the development of techniques that require extensive experimentation; and the sentences have both syntactic variety and reasonably rich semantics. RM contains 5,190 separate utterances (3,990 testing, 1,200 training) of 2,845 distinct sentences (2,245 training, 600 testing). We have extracted several types of CDGs from annotations of the RM sentences and tested their generality using the 7,396 sentences in RM2 (out of the 8,173) that are in the resource management domain but are distinct from the RM sentences. We compare these CDGs to each other and to the conventional CDG described previously.

The corpus-based CDGs were created by extracting the allowable grammar relationships from the RM sentences that were annotated by language experts using the SENATOR annotation tool, a CGI (Common Gateway Interace) HTML script written in GNU C++ version 2.8.1 (White, 2000). We tested two major CDG variations: those derived directly from the RM sentences (Sentence CDGs) and those derived from simple template-expanded RM sentences (Template CDGs). For example, "List MIDPAC's deployments during ⟨date⟩" is a sentence containing a date template which allows any date representations. For these experiments, we focused on templates for dates, years, times, numbers, and latitude and longitude coordinates. Each template name identifies a sub-grammar which was produced by annotating the appropriate strings. We then annotated sentences containing the template names as if they were regular sentences. Approximately 25% of the 2,845 RM sentences were expanded with one or more templates.

Although annotating a corpus of sentences can be a labor intensive task, we used an iterative approach that is based on parsing using grammars with varying degrees of restrictiveness. A grammar can be made less restrictive by ignoring:

- lexical information associated with a role value's modifiee in the ARVPs,
- feature information of two role values in an ARVP not directly related based on their modifiee relations,
- syntactic information provided by two role values that are not directly related,
- specific feature information (e.g., semantics or subcategorization).

Initially, we bootstrapped the grammar by annotating a 200 sentence subset of the RM corpus and extracting a fairly general grammar from the annotations. Then using increasingly restrictive grammars at each iteration, we used the current grammar to identify sentences that required annotation and verified the parse information for sentences that succeeded. This iterative technique reduced the time required to build a CDG from about one year for the conventional CDG to around two months (White, 2000).

Several methods of extracting an ARV/ARVP grammar from sentences or template-extended sentences were investigated. The ARVPs are extracted differently for each method; whereas, the ARVs are extracted in the same manner regardless of the method. Recall that ARVs represent the set of observed role value assignments. In our implementation, each ARV includes: the label of the role value, the role to which the role value was assigned, the lexical category and feature values of the word containing the role, the relative position of the word and the role value's modifiee, and the modifiee's lexical category and feature values (modifiee constraints). We use modifiee constraints for ARVs regardless of extraction method because their use does not change the coverage of the extracted grammar and not using the information would significantly slow the parser (Harper et al., 1999a). Because the ARVP space is larger than the ARV space, we investigate six variations for extracting the pairs:

1. **Full Mod:** contains all grammar and feature value information for all pairs of role values from annotated sentences, as well as modifiee constraints. For a role value pair in a sentence to be considered valid during parsing with this grammar, it must match an ARVP extracted from the annotated sentences.
2. **Full:** like **Full Mod** except it does not impose modifiee constraints on a pair of role values during parsing.
3. **Feature Mod:** contains all grammar relations between all pairs of role values, but it considers feature and modifiee constraints only for pairs that are directly related by a modifiee link. During parsing, if a role value pair is related by a modifiee link, then a corresponding ARVP with full feature and modifiee information must appear in the grammar for it to be allowed. If the pair is not directly related, then an ARVP must be stored for the grammar relations, ignoring feature and modifiee constraint information.
4. **Feature:** like **Feature Mod** except it does not impose modifiee constraints on a pair of role values during parsing.
5. **Direct Mod:** stores only the grammar, feature, and modifiee information for those pairs of role

Table 1: Number of ARVs and ARVPs extracted for each RM grammar.

ARVP Variation	Sentence CDG	Template CDG	Percent Increase
Full Mod	270,034	408,912	51.43%
Full	165,480	200,792	21.34%
Feature Mod	49,468	56,758	14.74%
Feature	36,558	40,308	10.26%
Direct Mod	41,124	47,004	14.30%
Direct	28,214	30,554	8.29%
ARVs	4,424	4,648	5.06%

values that are directly related by a modifiee link. During parsing, if a role value pair is related by such a link, then a corresponding ARVP must appear in the grammar for it to be allowed. Any pair of role values not related by a modifiee link is allowed (an open-world assumption).

6. **Direct:** like **Direct Mod** except it does not impose modifiee constraints on a pair of role values during parsing.

Grammar sizes for these six grammars, extracted either directly from the 2,845 sentences or from the 2,845 sentences expanded with our sub-grammar templates, appear in Table 1. The largest grammars were derived using the **Full Mod** extraction method, with a fairly dramatic growth resulting from processing template-expanded sentences. The **Feature** and **Direct** variations are more manageable in size, even those derived from template-expanded sentences.

Size is not the only important consideration for a grammar. Other important issues are grammar generality and the impact of the grammar on the accuracy of selecting the correct sentence from the recognition lattice of a spoken utterance. After extracting the CDG grammars from the RM sentences and template-expanded sentences, we tested the generality of the extracted grammars by using each grammar to parse the 7,396 RM2 sentences. See the results in Table 2. The grammar with the greatest generality was the conventional CDG for the RM corpus; however, this grammar also has the unfortunate attribute of being quite ambiguous. The most generalizable of extracted grammars uses the **Direct** method on template-expanded sentences. In all cases, the template-expanded sentence grammars gave better coverage than their corresponding sentence-only grammars.

We have also used the extracted grammars to post-process word graphs created by the word graph compression algorithm of (Johnson and Harper, 1999) for the test utterances in the RM corpus. As was reported in (Johnson and Harper, 1999), the word-error rate of our HMM recognizer with an embedded word pair language model on the RM test set

Table 2: Number of successfully parsed sentences in RM2 using the conventional CDG and CDGs derived from sentences only or template-expanded sentences.

ARVP Variation	# Parsed with Sentence CDG	# Parsed with Template CDG
Full Mod	3,735 (50.50%)	4,461 (60.32%)
Full	4,509 (60.97%)	5,316 (71.88%)
Feature Mod	5,365 (72.54%)	5,927 (80.14%)
Feature	5,772 (78.04%)	6,208 (83.94%)
Direct Mod	5,464 (73.88%)	5,979 (80.84%)
Direct	5,931 (80.19%)	6,275 (84.82%)
Conventional	7,144 (96.59%)	not applicable

of 1200 utterances was 5.0%, the 1-best sentence accuracy was 72.1%, and the word graph coverage accuracy was 95.1%. Also, the average uncompressed word graph size was 75.15 nodes, and our compression algorithm resulted in a average word graph size of 28.62 word nodes. When parsing the word graph, the probability associated with a word node can either represent its acoustic score or a combination of its acoustic and stochastic grammar score. We use the acoustic score because (Johnson and Harper, 1999) showed that by using a word node's acoustic score alone when extracting the top sentence candidate after parsing gave a 4% higher sentence accuracy.

For the parsing experiments, we processed the 1,080 word graphs produced for the RM test set that contained 50 or fewer word nodes after compression (out of 1,200 total) in order to efficiently compare the 12 ARV/ARVP CDG grammars and the conventional CDG (the larger word graphs require significant time and space to parse using the conventional CDG). These 1,080 word graphs contain 24.95 word nodes on average with a standard deviation (SD) of 10.80, and result in 1-best sentence accuracy was 75% before parsing. The number of role values prior to binary constraint propagation differ across the grammars with an average (and SD) for the conventional grammar of 504.99 (442.00), for the sentence-only grammars of 133.37 (119.48), and for the template-expanded grammars of 157.87 (145.16). Table 3 shows the word graph parsing speed and the path, node, and role value (RV) ambiguity after parsing; Table 4 shows the sentence accuracy and the accuracy and percent correct for words. Note that percent correct words is calculated using $\frac{N-D-S}{N}$ and word accuracy using $\frac{N-D-S-I}{N}$, where N is the number of words, D is the number of deletions, S is the number of substitutions, and I is the number of insertions.

The most selective RM sentence grammar, **Full Mod**, achieves the highest sentence accuracy, but at a cost of a greater average parsing time than the other RM sentence grammars. Higher accu-

ARVP Variation	Parse Time (sec.)	No. Paths	No. Nodes	No. RVs
Full Mod	33.89 (41.12)	2.21 (1.74)	10.59 (3.44)	19.51 (8.32)
Template Full Mod	41.85 (51.75)	2.78 (3.75)	10.76 (3.64)	19.93 (8.76)
Full	29.73 (36.68)	2.83 (2.92)	10.87 (3.54)	20.32 (8.86)
Template Full	36.80 (46.90)	3.40 (5.19)	11.03 (3.74)	20.77 (9.47)
Feature Mod	11.46 (14.46)	3.9 (5.97)	11.20 (3.94)	21.43 (10.49)
Template Feature Mod	13.80 (18.47)	4.22 (6.93)	11.28 (4.06)	21.81 (11.17)
Feature	11.60 (14.97)	5.19 (8.36)	11.72 (4.22)	23.41 (12.72)
Template Feature	14.24 (19.63)	6.86 (14.83)	11.94 (4.52)	24.47 (14.41)
Direct Mod	13.93 (19.73)	4.25 (6.49)	11.46 (4.27)	22.79 (13.44)
Template Direct Mod	17.28 (26.56)	4.62 (8.61)	11.45 (4.28)	22.95 (13.34)
Direct	19.95 (36.89)	8.08 (18.52)	12.81 (5.73)	32.85 (34.65)
Template Direct	28.02 (69.50)	9.98 (25.52)	12.95 (5.95)	33.36 (35.66)
Coventional	83.48 (167.51)	51.33 (132.43)	17.14 (8.02)	77.19 (76.26)

Table 3: Average parse times (SD), number of paths (SD), number of nodes (SD), and number of role values (SD) remaining after parsing the 1,080 word graphs of 50 or fewer word nodes produced for the RM test set using the 13 CDGs.

ARVP Variation	Sentence Accuracy	% Correct Words	Word Accuracy
Full Mod	91.94%	98.55%	98.19%
Template Full Mod	91.57%	98.50%	98.14%
Full	91.57%	98.49%	98.11%
Template Full	91.20%	98.45%	98.05%
Feature Mod	90.56%	98.38%	97.95%
Template Feature Mod	90.19%	98.34%	97.90%
Feature	90.28%	98.35%	97.91%
Template Feature	89.91%	98.29%	97.85%
Direct Mod	90.46%	98.37%	97.91%
Template Direct Mod	90.09%	98.32%	97.86%
Direct	89.91%	98.30%	97.82%
Template Direct	89.44%	98.25%	97.75%
Conventional	81.20%	97.11%	96.10%

Table 4: The sentence accuracy, percent correct words, and word accuracy from parsing 1,080 word graphs of 50 or fewer word nodes produced for the RM test set using the 13 CDGs.

racy appears to be correlated with the ability of the constraints to eliminate word nodes from the word graph during parsing. The least restrictive sentence grammar, **Direct**, is less accurate than the other sentence grammars and offers an intermediate speed of parsing, most likely due to the increased ambiguity in the parsing space. The fastest grammar was the **Feature-Mod** grammar, which also offers an intermediate level of accuracy. Its size (even with templates), restrictiveness, and speed make it very attractive. The template versions of each grammar showed a slight increase in average parse times (from processing a larger number of role values) and a slight decrease in parsing accuracy. The conventional grammar was the least competitive of the grammars both in speed and in accuracy.

5 Conclusion and Future Directions

The ability to extract ARV/ARVP grammars with varying degrees of specificity provides us with the ability to rapidly develop a grammar with the ability to improve sentence accuracy of our speech system. To achieve balance between precision and coverage of our corpus-induced grammars, we have expanded the RM sentences with templates for expressions like dates and times. The grammars extracted from these expanded sentences gave increased RM2 coverage without sacrificing even 1% of the sentence accuracy. We are currently expanding the number of templates in our grammar in an attempt to obtain full coverage of the RM2 corpus using only template-expanded RM sentences. We have recently added ten semantic templates to the grammar and have improved the coverage by 9.19% without losing any sentence accuracy. We are also developing a stochastic version of CDG that uses a statistical ARV, which is similar to a supertag (Srinivas, 1996).

References

H. U. Block. 1997. Language components in VERBMOBIL. In *Proc. of the Int. Conf. of Acoustics, Speech, and Signal Proc.*, pages 79–82.

Defense Advanced Research Projects Agency (DARPA). 1990. Extended resource management: Continuous speech speaker-dependent corpus (RM2). CD-ROM. NIST Speech Discs 3-1.2 and 3-2.2.

Entropic Cambridge Research Laboratory, Ltd., 1997. *HTK: Hidden Markov Model Toolkit V2.1.*

M. P. Harper and R. A. Helzerman. 1995. Extensions to constraint dependency parsing for spoken language processing. *Computer Speech and Language*, 9:187–234.

M. P. Harper, L. H. Jamieson, C. D. Mitchell, G. Ying, S. Potisuk, P. N. Srinivasan, R. Chen, C. B. Zoltowski, L. L. McPheters, B. Pellom, and R. A. Helzerman. 1994. Integrating language models with speech recognition. In *Proc. of the AAAI Workshop on the Integration of Natural Language and Speech Processing*, pages 139–146.

M. P. Harper, S. A. Hockema, and C. M. White. 1999a. Enhanced constraint dependency grammar parsers. In *Proc. of the IASTED Int. Conf. on Artificial Intelligence and Soft Computing.*

M. P. Harper, M. T. Johnson, L. H. Jamieson, and C. M. White. 1999b. Interfacing a CDG parser with an HMM word recognizer using word graphs. In *Proc. of the Int. Conf. of Acoustics, Speech, and Signal Proc.*

R. A. Helzerman and M. P. Harper. 1996. MUSE CSP: An extension to the constraint satisfaction problem. *Journal of Artificial Intelligence Research*, 5:239–288.

P. Jeanrenaud, E. Eide, U. Chaudhari, J. McDonough, K. Ng, M. Siu, and H. Gish. 1995. Reducing word error rate on conversational speech from the Switchboard corpus. In *Proc. of the Int. Conf. of Acoustics, Speech, and Signal Proc.*, pages 53–56.

M. T. Johnson and M. P. Harper. 1999. Near minimal weighted word graphs for post-processing speech. In *1999 Int. Workshop on Automatic Speech Recognition and Understanding.*

H. Maruyama. 1990a. Constraint Dependency Grammar and its weak generative capacity. *Computer Software.*

H. Maruyama. 1990b. Structural disambiguation with constraint propagation. In *Proc. of the Annual Meeting of Association for Computational Linguistics*, pages 31–38.

W. Menzel. 1994. Parsing of spoken language under time constraints. In *11th European Conf. on Artificial Intelligence*, pages 560–564.

W. Menzel. 1995. Robust processing of natural language. In *Proc. of the 19th Annual German Conf. on Artificial Intelligence.*

B. Natarajan. 1989. On learning sets and functions. *Machine Learning*, 4(1).

H. Ney, U. Essen, and R. Kneser. 1994. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language*, 8:1–38.

R. Pieraccini and E. Levin. 1992. Stochastic representation of semantic structure for speech understanding. *Speech Communication*, 11:283–288.

P. Placeway, R. Schwartz, P. Fung, and L. Nguyen. 1993. The estimation of powerful language models from small and large corpora. In *Proc. of the Int. Conf. of Acoustics, Speech, and Signal Proc.*, pages 33–36.

P. J. Price, W. Fischer, J. Bernstein, and D. Pallett. 1988. A database for continuous speech recognition in a 1000-word domain. In *Proc. of the Int. Conf. of Acoustics, Speech, and Signal Proc.*, pages 651–654.

L. A. Schmid. 1994. Parsing word graphs using a linguistic grammar and a statistical language model. In *Proc. of the Int. Conf. of Acoustics, Speech, and Signal Proc.*, pages 41–44.

S. Seneff. 1992. TINA: A natural language system for spoken language applications. *American Journal of Computational Linguistics*, 18:61–86.

B. Srinivas. 1996. 'Almost parsing' technique for language modeling. In *Proc. of the Int. Conf. on Spoken Language Processing*, pages 1173–1176.

L. G. Valiant. 1984. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.

C. M. White. 2000. *Rapid Grammar Development and Parsing Using Constraint Dependency Grammars with Abstract Role Values.* Ph.D. thesis, Purdue University, School of Electrical and Computer Engineering, West Lafayette, IN.

J. H. Wright, G. J. F. Jones, and H. Lloyd-Thomas. 1994. Robust language model incorporating a substring parser and extended N-grams. In *Proc. of the Int. Conf. of Acoustics, Speech, and Signal Proc.*, pages 361–364.

S. J. Young, N. H. Russell, and J. H. S. Thornton. 1989. Token passing : a simple conceptual model for connected speech recognition systems. Technical Report TR38, Cambridge University, Cambridge, England.

S. J. Young, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, 1997. *The HTK Book.* Entropic Cambridge Research Laboratory Ltd., 2.1 edition.

V. Zue, J. Glass, D. Goodine, H. Leung, M. Phillips, J. Polifroni, and S. Seneff. 1991. Integration of speech recognition and natural language processing in the MIT Voyager system. In *Proc. of the Int. Conf. of Acoustics, Speech, and Signal Proc.*, pages 713–716.