INTERFACING A CDG PARSER WITH AN HMM WORD RECOGNIZER USING WORD GRAPHS

M. P. Harper, M. T. Johnson, L. H. Jamieson, S. A. Hockema, and C. M. White

Purdue University, School of Electrical and Computer Engineering
West Lafayette, IN 47907
{harper,mjohnson,lhj,hockema,robot}@ecn.purdue.edu

ABSTRACT

In this paper, we describe a prototype spoken language system that loosely integrates a speech recognition component based on hidden Markov models with a constraint dependency grammar (CDG) parser using a word graph to pass sentence candidates between the two modules. This loosely coupled system was able to improve the sentence selection accuracy and concept accuracy over the level achieved by the acoustic module with a stochastic grammar. Timing profiles suggest that a tighter coupling of the modules could reduce parsing times of the system, as could the development of better acoustic models and tighter parsing constraints for conjunctions.

1. INTRODUCTION

In this paper, we describe a prototype of a spoken language system that integrates a speech recognition component based on hidden Markov models with a constraint dependency grammar (CDG) parser. The underlying goal of our combined system is to identify the 'best' overall sentence candidate with respect to all available knowledge sources and map that candidate to an internal representation. How best to achieve this goal is an important problem that deserves careful study.

The question of how to integrate the language models with speech recognition systems is gaining in importance as speech recognizers are increasingly used in human/computer interfaces and dialog systems [3, 10]. Although many current systems tightly integrate stochastic language models, with a power limited to a regular grammar, into the recognizer, a language processing module is needed to ensure that the speech signal is mapped to an internal representation that the computer requires in order to act based on the spoken interaction with the user. Obtaining a syntactic representation for the spoken utterance, therefore, has a high degree of utility for mapping to a semantic representation.

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 [3]. A loosely-integrated language model can be developed independently of the speech recognition component, which is clearly an advantage. However, such a module cannot directly reduce the search space of the acoustic module. On the other hand, 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. A moderately integrated language model falls in between the previous two in that it can guide the acoustic module's search; however, moderate integration is likely to be more difficult to engineer because information must travel between the modules in both directions.

To begin investigating the integration of the speech recognition component with the CDG parser, we built a loosely coupled system and conducted experiments to determine the best way to enhance the processing speed and accuracy of the combined system. Figure 1 depicts the system (dashed lines indicate components not incorporated in the current system). A word graph is used to pass potential sentence hypotheses between the acoustic and the language modules. This representation was chosen because it allows us to investigate the potential for the CDG module to impact the choice of the final sentence candidate in a more tightly integrated system. In addition, the word graph is a more compact datastructure than the recognition lattice of the speech recognizer. Word graphs support an aggregate processing view as well as a stream-based view, providing flexibility for examining integration alternatives.

2. THE MODULES

In this section we describe the components of our system.

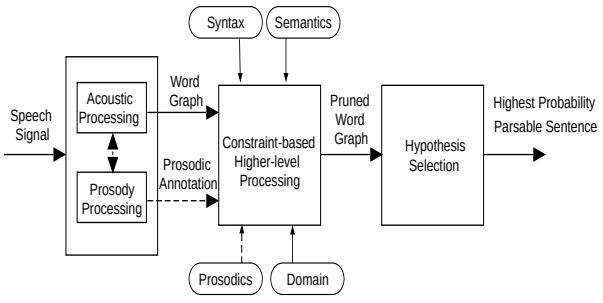

Figure 1: Block diagram of the loosely-coupled spoken language system.

2.1. The Word Recognizer

The underlying structure for the recognition network is provided by a multiple-mixture triphone Hidden Markov Model (HMM) [9], with a simple integrated grammar (either bi-gram or word pair). Our system was implemented using HTK Version 2.1 by Entropic [6, 12]. Observation output distributions under this system are represented by multiple stream Gaussian Mixture Models (GMMs), with distribution parameters and state transition probabilities re-estimated using the Baum-Welch algorithm. Recognition is achieved using a token-passing implementation of the Viterbi algorithm, the output of which is a large recognition lattice containing acoustic and grammar likelihoods for each word node.

The output of our recognition module is a word graph annotated with probabilities that allow the highest probability sentences to be examined in order of decreasing probability. To accommodate the language model, we define a word graph as a directed acyclic graph representing the possible word paths through the utterance such that the nodes represent the words and connecting arcs represent word transitions. Construction of the word graph is accomplished by post-processing the recognition lattice. Since the lattice contains full alignment information such as start and end times for each word node, it can include many identical or nearly identical paths that vary only with regard to time alignment. The language processing system does not need this alignment information and is slowed by the redundant information. Word graph generation, therefore, includes an algorithm to eliminate identical sub-graphs from the lattice, resulting in a graph that represents all possible word-level paths without eliminating or adding any path possibilities.

Each word graph has a distinct starting node and a distinct ending node. Pseudocode for the compression algorithm is shown below, where `prevlist(i)` denotes the list of nodes that directly precede node i, and `nextlist(i)` denotes the list of nodes that directly follow i.

```
While (number of graph nodes continues to change)
    For all Nodes i
        For all Nodes j ≠ i
            If ((prevlist(i)=prevlist(j)) OR
                (nextlist(i)=nextlist(j))
               Replace all references to j with i
               Delete node j
```

The word graphs created using this algorithm have a size that is on average 42.8% of original lattice size, and the time to create the word graph represents only 1.9% of recognition time. An alternative algorithm was initially considered that compressed nodes based on time alignment. For comparison, this algorithm created word graphs with an average size of 42.4% of original lattice size, and the time to create the word graph represented only 1.1% of the recognition time. Although this algorithm was able to construct more compact word graphs (0.4% smaller), it did so at the expense of adding spurious paths not found in the original recognition lattice. The first algorithm was chosen because it creates only slightly larger word graphs containing only legal lattice paths.

We have explored the effects of different types of pruning and stochastic grammars on word graphs and N-best lists produced by our acoustic module trained on the RM corpus [5]. It was determined that by allowing an unlimited number of active models in our HMM that the word graph contains the correct sentence in 98% of the test cases. In 17% of the cases, the word graph contains the correct sentence but the correct sentence is not the top candidate in an N-best search; in .89% of the cases, the word graph contained the correct sentence when it was not in the 10-best list. The average size of these word graphs was 23.3 words, compared to an average actual sentence length of 8.8. It was concluded that the word graphs form an efficient interface between continuous-speech recognition and our parser: they achieve high coverage with modest graph densities, and they can be generated with computation times comparable to standard N-best recognition without introducing spurious sentence hypotheses.

2.2. The Parser

Our language module is based on Constraint Dependency Grammar, which was originally conceived by Maruyama [7]. In contrast to context-free grammars, CDG uses constraints rather than production rules for its grammar rules, and the sentence structure is recorded by assigning role values consisting of dependency links (called modifiees) and tags (called labels) to named slots (called roles) associated with each word. The assignment of a role value to a role specifies a grammar relation that is a component of the parse for the sentence. For example, when the governor role for a noun is assigned a role value subject-3, this indicates that it is a subject governed by the word in the sentence with a position tag 3. For CDG, the parsing algorithm is framed as a constraint satisfaction problem in which the parsing rules are the constraints and the solutions are the parses. A sentence s is said to be **generated** by the grammar G if there exists an assignment **A** that maps role values to the roles for each of the words such that the constraints are satisfied. CDG has been adapted to support the simultaneous analysis of sentences with multiple alternative lexical categories (e.g., *can* is a noun, verb, or modal) and multiple feature values (e.g., *the* as a determiner can modify nouns that are third person singular or third person plural) [3]. The resulting parser can also simultaneously process the sentence hypotheses resulting from the word segmentation ambiguity of a speech recognizer [3].

The CDG parsing algorithm offers a flexible and powerful parsing framework for our spoken language system. In addition to the fact that it can simultaneously analyze all sentences in a word graph [3], there are a variety of other features that make it attractive. First, the generative capacity of a CDG is beyond context-free languages [7]. There is evidence for the need to develop parsers for grammars that are more expressive than the class of context-free grammars but less expressive than context-sensitive grammars. Second, free-order languages can be handled by a CDG parser without enumerating all permutations because order among constituents is not a requirement of the grammatical formalism [3]. Third, the CDG parser uses sets of constraints which operate on role values assigned to roles to determine whether or not a string of terminals is in the

grammar. These constraints can be used to express legal syntactic, prosodic, semantic relations, as well as context-dependent relations. Constraints can be ordered for efficiency or they can be withheld. The presence of ambiguity can trigger the propagation of additional constraints to further refine the parse for a sentence [2]. This flexibility can be utilized to create a smart language processing system: one that decides when and how to use its constraints based on the state of the parse. Fourth, a CDG parser is highly parallelizable [4]. Finally, we have developed methods for learning CDGs directly from labelled sentences in a corpus [11]. This work was enabled by the fact that CDG constraints are PAC learnable from positive examples of dependency relations. This work provides a foundation for speeding the grammar development cycle and for creating constraints from corpora that will be tightly constraining.

3. EXPERIMENTAL EVALUATION

3.1. The Corpus and the CDG Grammar

In order to evaluate the use of word graphs as an effective interface between our speech recognition and natural language components, we have performed initial experiments on the DARPA Resource Management (RM) corpus [8], a 1000-word task domain defined for speech recognition evaluation. It was chosen for several reasons: the properties of the corpus are fairly well understood; its manageable size (5000 utterances) makes it a good platform for the development of techniques allowing extensive experimentation; and it is a corpus with syntactic variety and reasonably rich semantics.

For this experiment, we have constructed a CDG to parse the sentences in this corpus. The corpus contains 2845 sentences derived from sentence templates based on interviews with naval personnel familiar with naval resource management tasks. It contains wh-questions and yes/no-questions about ships, ports, etc., along with commands to control a graphics display system. While the RM corpus is restricted, it does contain a wide variety of grammar constructs and a fairly rich semantics. The CDG constructed to cover this corpus uses 16 lexical categories, 4 roles, 32 labels, 15 lexical feature types (for example, `subcat`, `agr`, `case`, `verb_type` (e.g., progressive), `mood`, `gap`, `inverted`, `voice`, `gender`, `sem_type`, and `conj_type`), and a total of 1384 unary and binary constraints.

3.2. Results

To evaluate the impact of the word graph construction scheme on the running time of our parser, we initially compared the sizes and parsing times of word graphs directly constructed (i.e., without compression) from the recognition lattice with the word graphs constructed using the compression algorithm from section 2.1. Because extensive investigation of uncompressed word graph parsing has a high processing cost, we limited the investigation to five randomly selected uncompressed word graphs and their compressed counterparts. In every case the resulting word graph was at most 52% the size of the uncompressed word graph (with an average of 44%), and the parsing time for the compressed word graphs was at least 3.3 times faster (with an average of 4.3 times faster).

The recognizer with stochastic grammar returns the top sentence candidate in 83% of the cases; hence, 17% of the test utterances are not correctly selected as the target sentence. To evaluate the impact that the CDG grammar has on recognition accuracy of the combined system, we have conducted two experiments to determine whether the addition of the parsing module improves the sentence selection accuracy of the combined system over the recognizer with stochastic grammar. In the first experiment, word graphs were generated by the speech recognizer for 100 randomly selected utterances and then parsed by our CDG parser. In the second experiment, word graphs were generated by the speech recognizer for 103 utterances chosen randomly from those for which the top scoring hypothesis was *not* the target sentence.

In the first experiment, the target sentence had the highest probability in 90 out of the 100 utterances tested, so the recognizer with stochastic grammar achieved a 90% sentence selection accuracy. Furthermore, in the ten cases that the recognizer plus stochastic grammar failed to return the target sentence, four of the returned sentences differed only slightly from the meaning of the target sentence (substitution or deletion of a determiner), resulting in a concept accuracy [1] of 94%. The word graphs constructed by this recognizer placed the target sentence as the highest scoring path in 90 of the word graphs, second highest in nine, and third highest in one. For the 10 word graphs in which the top scoring hypothesis was not the target sentence, the CDG constraints eliminated the higher scoring sentence hypotheses and returned the target sentence in 4 additional cases. Hence, our CDG parser improves the sentence accuracy from 90% to 94% on this set of sentences. Furthermore, in the six cases that the the parser failed to return the target sentence, four of the sentences returned differed only slightly from the meaning of the target sentence, giving a concept accuracy of 98% compared to 94% for the recognizer.

In the second experiment, the target sentence was not identified by the recognizer plus stochastic grammar as the highest scoring sentence hypothesis in any of the 103 test utterances; however, 42 of the sentences returned differed only slightly from the target sentence, giving a sentence selection accuracy of 0% and a concept accuracy of 41%. The word graphs constructed by this recognizer placed the target sentence as the the second highest scoring path in 70 of the graphs, third highest in 15 of the word graphs, fourth highest in 13 of the word graphs, fifth in two, sixth in one, and seventh in two. When these word graphs were parsed by the CDG parser, the target sentence was correctly selected in 38 cases, increasing the sentence selection accuracy from 0% for the recognizer to 37%. In the 65 cases that the parser plus recognizer failed to return the target sentence, 50 of the sentences returned differed only slightly from the target sentence. Hence, the concept accuracy of our combined system was 85% compared to 41% for the recognizer plus stochastic grammar. In the remaining fifteen cases, the parser selected sentences that were not semantically close to the target utterance. Most of these cases involved word confusibility (e.g., *get* vs. *give*; *in* vs. *and*; *sixty* vs. *sixteen*); however, some had multiple word differences.

These experiments show that compact word graphs can

be constructed for the test utterances in the RM corpus, and that a hand-constructed CDG grammar provides additional constraints that improve the sentence selection accuracy. In addition to increasing the sentence selection accuracy, the CDG parser was able to eliminate many spurious sentence candidates. A measure of this is the percent decrease in the size of the word graph as a result of applying the grammar constraints. The average word graph size before parsing in experiments 1 and 2 was 21.83 and 26.48 word nodes, respectively. After parsing, the word graphs contained 15.62 and 18.66 word nodes, respectively.

These results confirm that constraints improve accuracy, but they do not necessarily imply that a loosely integrated system with a word graph as the interface represents the most efficient integration of the acoustic module with syntactic and semantic constraints. To get a better idea, we have compared the time to parse the word graphs from our two experiments with the time to parse sentences, one by one in order of acoustic probability until the target sentence is reached. Although the word graph parser is sometimes faster than this one-at-a-time parser, the time to process the list of sentences takes only 25% of the time that it takes to process the word graph in first experiment and 29% of the time in the second. These results suggest that a more moderate coupling should be more time efficient. At the very least, methods for decreasing word graph size with minimal reduction in accuracy will improve parsing time: the sizes of the word graphs are significantly correlated with the time to parse the word graph (r=.65 and r=.71).

This is not the entire story, however. A second factor was found to impact the parsing time of the word graphs: the presence of one or more conjunctions in the word graphs. The time to parse word graphs without conjunctions was 11% and 12% of the time to parse word graphs with conjunctions in experiments 1 and 2, respectively. Although the word graphs with conjunctions were larger than the word graphs without conjunctions (15.5 versus 25.9 and 19.21 versus 30.05), this does not completely account for the extreme difference in parsing time. There are two additional factors. First, word graphs with conjunctions tend to have significantly more role values per node at the outset than word graphs without conjunctions, namely 108.16 versus 207.29 and 155.92 versus 259.87; hence, conjunctions create a higher degree of ambiguity for the parser to deal with. This suggests the need for better acoustic models for conjunctions to speed up the processing of the word graphs. A second factor concerns the fact that there was a disparity in the number of role values that remain in the parse graph but are not used in parses of word graphs with and without conjunctions. The number of role values remaining at the end of parsing not used in parses was .08 (only 3 cases) and 2.15 (4 cases) for word graphs without conjunction and 12.56 (24 cases) and 25.07 (46 cases) on average in the presence of a conjunction. This suggests that grammar constraints constructed to parse conjunctions were not as restrictive as the other constraints.

4. CONCLUSIONS AND FUTURE WORK

Given the results of this experimental evaluation, we conclude that syntactic and semantic constraints improve sentence recognition and concept accuracy over the word recognizer with stochastic grammar. Constraints reduce word graph size, suggesting that they are capable of reducing the search space for the top sentence candidate in a more tightly integrated system. Finally, more feedback from the parser to the recognizer is likely to increase not only the accuracy, but also the speed of the combined system.

In light of our investigations, future work will focus on three aspects of the problem of integrating the acoustic and language models. Grammar constraints for future experiments will be learned directly from corpora to obtain tighter constraints. A moderately coupled integration scheme will be designed to improve the efficiency of the combined system. Finally, because conjunctions are a source of significant parser ambiguity, more selective acoustic models for conjunctions (such as *and*) will be developed.

5. ACKNOWLEDGMENTS

This research is supported by the National Science Foundation under Grant No. IRI-9704358.

6. REFERENCES

[1] M. Boros, W. Eckert, F. Gallwitz, G. Goerz, G. Hanrieder, and H. Niemann. Towards understanding spontaneous speech: Word accuracy vs. concept accuracy. In *Proc. of ICSLP*, volume 2, pages 1009–1012, 1996.

[2] M. P. Harper and R. A. Helzerman. Managing multiple knowledge sources in constraint-based parsing spoken language. *Fundamenta Informaticae*, 23(2,3,4):303–353, 1995.

[3] Mary P. Harper and Randall A. Helzerman. Extensions to constraint dependency parsing for spoken language processing. *Computer Speech and Language*, pages 187–234, 1995.

[4] R. A. Helzerman and M. P. Harper. Log time parsing on the MasPar MP-1. In *Proc. of the International Conference on Parallel Processing*, volume 2, pages 209–217, 1992.

[5] M. T. Johnson, M. P. Harper, and L. H. Jamieson. Effectiveness of word graphs for interfacing speech recognition and language models. In *Proc. of ICSLP*, 1998.

[6] Entropic Cambridge Research Laboratory Ltd. HTK version 2.1, 1997.

[7] H. Maruyama. Structural disambiguation with constraint propagation. In *Proc. of the Assoc. for Comp. Ling.*, pages 31–38, 1990.

[8] P. J. Price, W. Fischer, J. Bernstein, and D. Pallett. A database for continuous speech recognition in a 1000-word domain. In *Proc. of ICASSP*, volume 1, pages 651–654, 1988.

[9] Lawrence R. Rabiner. Tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.

[10] Ludwig A. Schmid. Parsing word graphs using a linguistic grammar and a statistical language model. *Proc. of ICASSP*, 2:41–44, 1994.

[11] C. M. White, M. P. Harper, T. Lane, and R. A. Helzerman. Inductive learning of abstract role values derived from a constraint dependency grammar. In *Proc. of the Automata Induction, Grammatical Inference, and Language Acquisition Workshop*, July 1997.

[12] Steve Young, Julian Odell, Dave Ollason, Valtcho Valtchev, and Phil Woodland. *The HTK Book*. Entropic Cambridge Research Laboratory Ltd., 2.1 edition, 1997.