Towards Habitable Systems: Use of World Knowledge to Dynamically Constrain Speech Recognition

Sheryl R. Young and Wayne H. Ward

October 1988
CMU-CS-88-184

Abstract

Current state-of-the art speaker-independent continuous speech recognizers are able to achieve word recognition rates well above 90 percent with lexicons of 1000 words or less using grammars with perplexity 60 or less. Performance of these systems decreases rapidly as the perplexity of the grammar increases. As we allow users more flexibility in interacting with recognition systems, the size of the lexicons and perplexity of the grammars increase greatly. Allowing spontaneous speech instead of read speech compounds the problems even more. Other sources of knowledge may be available to help constrain the ever more complex search spaces in such systems. When recognition systems are used in performing problem solving tasks, predictable features of the user's behavior can be used to aid recognition. We describe a system (MINDS) which uses additional constraints based on dialog interactions. The constraints are applied in a manner that allows optimum performance when users behave predictably, and degrades gracefully when they do not. We also present an evaluation of the system's performance to show the utility of the additional knowledge sources.

Appeared in *The Second Symposium on Advanced Man Machine Interface Through Spoken Language, Oahu, Hawaii, Nov. 19-22, 1988*

This research was supported by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 5167, under contract number N00039-85-C-0163. The authors wish to thank Alexander Hauptmann, Edward T. Smith, Philip Werner and D. Raj Reddy for their contributions to this research. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U S Government.

1. Overview

One of the biggest problems in computer speech recognition is coping with large search spaces. The search space for speech recognition contains all the patterns associated with words in the lexicon as well as all the legal word sequences. The most widely used recognition systems are hidden Markov model (HMM) based. In these systems, typically, each word is represented as a sequence of phonemes, and each phoneme is associated with a sequence of markov states. As search space size decreases, recognition performance increases. Knowledge can be used to constrain the exponential growth of a search space and hence increase processing speed and recognition accuracy [17, 6, 11]. Currently, the most common approach to constraining search space is to use a grammar. The grammars used for speech recognition dictate legal word sequences. Normally they are used in a strict left to right fashion and embody syntactic and semantic constraints on individual sentences. These constraints are represented in some form of probabilistic or semantic network which does not change from utterance to utterance [12, 3, 11].

As we move toward habitable systems and spontaneous speech, the search space problem is greatly magnified. Habitable systems permit system users to speak naturally. Grammars which try to cover even naturally elicited syntactically accurate sentences have perplexities that are an order of magnitude larger than the perplexities of grammars typically used by speech recognizers. Spontaneous speech grammars will have even larger perplexities. Spontaneous speech is ungrammatical and contains much editing. These edits can occur anywhere within a sentence and are often preceeded by interjections. Additionally, spontaneous speech exhibits human noise in the form of filled pauses. Finally, the above phenomena are compounded by the presence of multiple sentences uttered without pausing at sentence boundaries and silent pauses within incomplete phrases. Hence, the notion of a well formed sentence exhibiting typical syntactic regularities is not applicable when processing spontaneous speech. These problems point to the need of using knowledge sources beyond typical syntax and semantics to constrain the pattern matching process in speech recognition.

There are many other knowledge sources besides syntax and semantics. Typically, these are clustered into the category of pragmatic knowledge. Pragmatic knowledge minimally includes inferring plans, using context across clausal and sentence boundaries, determining local and global constraints on utterances and dealing with definite and pronominal reference. Work in the natural language community has shown that pragmatic knowledge sources are important for understanding language. People communicate to accomplish goals, and the structure of the plans to accomplish them are well understood [18, 19, 7, 20, 21] [5, 1, 16, 9] [4]. When speech is used in a structured task such as problem solving, pragmatic knowledge sources are available for constraining search spaces.

In the past, pragmatic, dialog level knowledge sources were used in speech to either correct speech recognition errors [8, 2] or to disambiguate spoken input and perform inferences required for understanding [12, 15, 14]. In these systems, pragmatic knowledge was applied to the output of the recognizer.

In this manuscript we describe an approach for flexibly using contextual constraints to dynamically circumscribe the search space for words which can be matched against a speech signal. We use pragmatic knowledge to derive constraints about what the user is likely to say next. Then we loosen the constraints in a principled manner. Hence, we generate sets of predictions which range from very specific to very general ("layered predictions"). To enable the speech system to give priority to recognizing what a user is most likely to say, each prediction set dynamically generates a grammar which is used by the speech recognizer. The prediction sets are tried in order of most specific first, until an acceptable parse is found. This allows optimum performance when users behave predictably, and displays graceful degradation when they do not. The implemented system (MINDS) uses these layered constraints to guide the search for words in our speech recognizer. For our recognizer, we use a modified version of the SPHINX (Lee, 1988) large vocabulary, speaker independent, continuous speech recognition system.

The following section places the research described in the context of the overall MINDS system architecture. The following two sections describe the methods used to generate predictions and use them to guide recognition. We then present the results of two studies which illustrate both perplexity reduction and performance improvements resulting from the use of predictions.

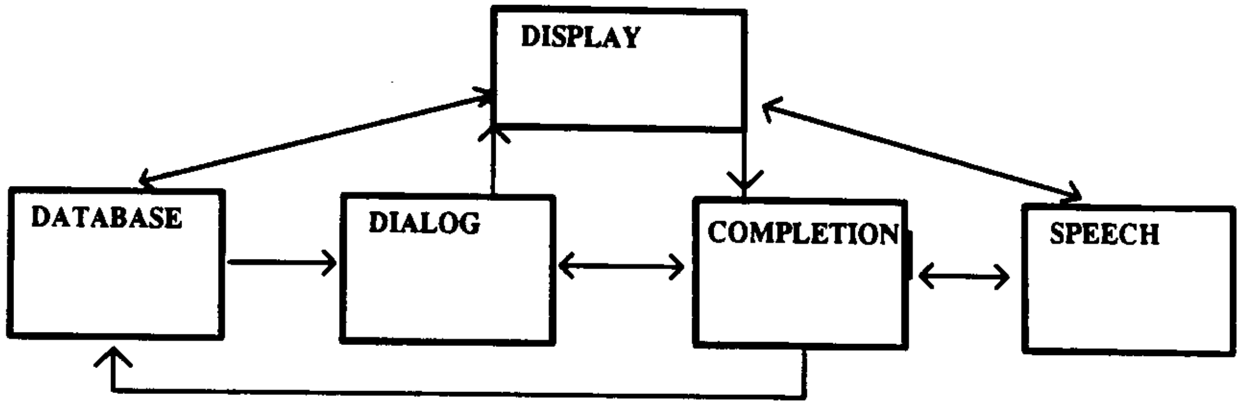
Finally, we describe our current work in progress.

2. MINDS System Architecture

The MINDS system uses pragmatic knowledge sources predictively to circumscribe the search space for words in a speech signal [10, 22]. The pragmatic knowledge sources are embodied in an elaborate dialog model. The dialog model infers plans, performs plan tracking, deals with clarification subdialogs and dynamically computes constraints using local and global focus, or contextual information propagated from prior information seeking stages. To allow for diverse user behavior, MINDS uses a principled, general algorithm for relaxing constraints. Constraints are organized into sets that are successively more general, called "**layers**". When some constraints are violated, we use the non-violated constraints to reduce search space. Additionally, the flexible use of constraints allows the use of knowledge sources that are less certain to be true. Users that behave consistently can benefit greatly from enhanced recognition and the system will show a graceful degradation on those who do not.

To enable the MINDS system to generate predictions and use them to guide the speech recognizer, we have partitioned the system into five interacting modules, as seen in Figure 1.

Figure 1: MINDS System Modules



The **speech module** is composed of a modified version of the SPHINX speaker independent, continuous, large vocabulary speech recognizer. This version of SPHINX uses finite state grammars to constrain search. The grammars are dynamically generated after each utterance by the dialog module and sent to the completion module. Hence, the speech module receives input from the completion module and the speaker. It sends its output to both the completion module and the display module.

The **completion module** is composed of a semantic parser, representations of the domain, the database, and the finite state grammar. The completion module communicates with the speech module, the database, and the dialog module. It takes the speech output, parses it and performs any necessary disambiguation. Then it takes its semantic representation and communicates it to the dialog module and generates a database query. Once the predictions are generated, the completion module indexes them into precompiled portions of the finite state, semantic grammar and places restrictions on the expansions of the rewrite rules embedded in the finite state nets. The nets are then merged.

The **dialog module** is composed of a domain knowledge base, a hierarchical representation of possible domain problem solving plans, and a set of heuristics for propagating constraints, inferring plans and tracking plans. The dialog module receives input from both the completion module and the database module so that it can track all information communicated. The dialog module is responsible for generating layered sets of predictions. It communicates these to the completion module so they can be expanded into potential surface forms.

The **database module** is composed of the Informix^{TM} relational database management system filled with a domain database, an "expert" interface to the database, and a natural language generator. The database module receives input queries from the completion module. If these are either ambiguous or computationally expensive, the "expert" interface has the option of querying other system modules or the user for clarification / further specification. The "expert" interface translates query inputs into a form necessary for the database. Additionally, it translates the output into a semantically meaningful form. The output is then sent to the dialog module while the natural language generator produces sentential output which is then communicated to the display module.

The **display module** is composed of four displays. Two displays are maps which display the current version of the world and can zoom into areas of interest to the system user. The third display depicts detailed information previously communicated in the dialog, while the fourth display is devoted to communicating with the user and all the system modules. The fourth display contains a type in window which also displays the generated natural language database response as well as the spoken utterance. Additionally, it contains windows for displaying clarifications requested by other system modules, and a window for displaying the test set perplexity of the just parsed utterance. This module communicates with all other modules and knows the complete system state.

When spoken information is input to the system, it is first processed by the speech module using the predictions generated earlier. Its output is sent to both the display module (where it can be corrected if necessary) and the completion module. The completion module performs a semantic parse on the information and generates a database query. The semantic parse is sent to the dialog module and the database query is sent to the database module. The dialog module then determines which possible plan steps were activated by the input and uses the database response to gather further context. It then generates a new set of layered predictions and passes these back to the completion module for expansion and use by the speech module. Each of the system modules described above run in a distributed environment.

In the next section we describe the use of plans to limit search space and the algorithms which enable the MINDS system to generate layered sets of predictions.

3. Plan Based Constraints: Prediction Generation

The idea underlying the MINDS system is that tracking all information communicated (user questions and database answers) enables a system to infer a set of possible problem solving plans and to track progress through these plans. In the convention of Newell and Simon (1972) these plans are represented as hierarchically organized goal states. *For example, in the domain of dealing with disabled ships, a goal state would be finding a replacement ship.* As each new input sentence is spoken, the system analyzes the utterance to determine the concepts expressed and uses these concepts to activate goal states. To derive plan based constraints on future utterances, active goal states are assessed to determine legal next states. *For example, when finding a replacement ship, some of the legal next states which follow a question about the ships in some region are questions about more ships in the region, questions about availability of these ships, and questions about the ships' equipment.* Because speech systems use grammars to guide word transitions, we associated a list of required and optional concepts with each goal state *(e.g. concepts associated with a goal state for ship equipment include equipment, weapons, aircraft, electronics, etc.).* The list of possible next states is used to generate a set of possible concepts which could be spoken in the next utterance. This set is then limited by local and global focus which takes into account prior context, rules about reference, etc. The speech recognizer only searches for surface forms expressing concepts in this set.

3.1. Layered Predictions

Plan based constraints are quite effective in reducing search space by delimiting the types of information likely to be communicated [10, 22]. But plan based constraints are based upon inferring user plans. Usually it is not possible to either definitively select a single plan step given an input utterance. Similarly, users may exhibit unexpected behavior by either violating the hierarchical nature of a plan or leaving plan steps incomplete. As both domain size increases and spontaneously generated speech is used for generating queries, these problems become magnified.

To overcome the problem of multiple active plans and unexpected user behavior, we instituted three procedures. First, we designed an algorithm to select "the best" plan step or goal state from the list of possible goal states activated by the preceeding utterance and database response. Here we preferred goal states that were both complete and most likely to follow given the previous goal states activated. Second, we maintained a list of all other active goal states, including those which were not hierarchically embedded. These activated states were used to generate some alternate predictions about what the user could say. Third, we generated sets of layered predictions about the content of the following utterance. The predictions ranged from very specific to very general. These layered predictions were rank ordered to reflect both amount of constraint provided as well the reliability of the knowledge sources used to generate them. It should be noted that the least constraining prediction layer allowed all domain concepts. This means that the system could cope with any statement the user might say even if its not included in the system grammar. However, the system cannot cope with words which are not included in the system lexicon.

By layering predictions, we allow the system to reparse a speech signal with a different grammar until such time as a good parse is received. The ability to reparse an utterance also enables us to use less reliable knowledge sources to further constrain our predictions. Hence, we added two additional knowledge sources to the system: user domain expertise models and preference orderings for conjunctive goals.

Observing that system users with significant domain expertise solved problems using very different plans than novice users, we attempted to model the effects of expertise by constructing domain knowledge models of novice, intermediate and expert system users. Our user models were represented as subsets of the domain knowledge base. The models differed primarily by the existence of relations between domain objects. *For example, an expert user would know that each class of ships has a set of default equipment and is suited for particular types of tasks, while a novice user might not be aware that shiptypes are divided into ship classes.* The user models were then used to construct control schemas which specified which goal states were exclusive. *To further the last example, a control schema for an expert user would show that if the user asked about a shipclass they would not ask about default equipment.* These models were hand coded from the training set data.

Similarly, we used the training data to derive probabilistic orderings on conjunctive subgoals. These orderings told us which conjunctive goals would be executed first, second, etc. The orderings were computed across individuals (although our training data only came from two people). However, there is no reason why these could not be automatically obtained for individual system users in future systems.

Thus, the MINDS system used the following knowledge sources to derive predictions about the content of a user's next utterance:

1. knowledge of problem solving plans represented as a hierarchical goals,
2. semantic knowledge about the application domain's objects, attributes and their interrelations (a domain knowledge base),
3. domain independent knowledge about methods of speaking, appropriateness of references and partial utterances (local and global focus)
4. dialog history knowledge about information previously communicated,
5. discrete models of user domain expertise as described above, and
6. information about user preferences for ordering conjunctive subgoals

These knowledge sources were used by the prediction module to perform iterative analyses of the dialog after each input/database response pair and generate sets of restrictions on the next utterance. The predictions generated are layered. Each successive layer is less constraining than the prior layer. The most constraining prediction set is generated using all knowledge sources listed above. The next set does not use user models and uses a larger non-overlapping set of goal states. Further sets are generated by moving upward in the goal hierarchy, allowing more plans to be executed. The prediction sets become successively more general, hence the term "layered". Ultimately, the entire system grammar will be used. If this fails, an "allword" recognition is attempted where any word sequences are allowed (providing of course that the words are in the system lexicon).

3.2. Derivation of Predictions

To illustrate how the information contained in a goal state or plan step is used to generate predictions, we present simplified, although prototypical representations of both a plan step and the control information which encodes our "less reliable" information about users. These are depicted in Figures 2 and 3, presectively.

Figure 2: Example Goal State Schema

```
[ Shipclass
  :Concepts-Required ((Shipclass single-use Child-restrictions*
                                            (knoxclass perryclass)))
                    Concept Times-used Restriction-pointers
  :Optional-Concepts ((Region single-use Child-restrictions*
                                            (Persian-gulf)))
  :Optional          (Not for expert-user)
                     True/Nil/User-consideration
  :Next-states       (Find-Replacement) goal-state
  :Parent            (Find-Replacement)
  :Children          (none)
  :Control           (none)
]
* = Computed by local and global context
```

Figure 3: Example Control Schema

```
[Control00030 - for Find-replacement
  :Exclusive ((Shipclass    Equipment))
  :Omit      (Shiptype)
  :Order     ((.90 Shipclass .10 Mission-Info)
              (.90 Mission-Info  .10 Shipclass))
]
```

The **concepts-required** and **optional-concepts** slot values are used to specify the concepts relevant when a user transits to the goal state. The number of concepts per goal state and the number of goal states a user could progress to next determine the size of the lexicon the speech recognition system must analyze. The **control** slot contains a pointer to a **control schema** wherever the child slot is not empty.

Control schemas predict whether any child states are likely to be omitted and any preferred orderings on the states for a specific system user. They are used to generate the most constraining prediction layers.

As seen in Figure 3, there are three slots in a control schema. The **order** slot stores information about preferred orderings among non-optional, conjunctive subgoal states. The **exclusive** slot stores pairs of goal states which are exclusive because the information in the first allows the user to infer the information in the second. The **omit** slot store a list of goal states the user omits because they are unaware of the domain concepts.

Control schemas are attached to parent goal states to predict which child states will be visited. Hence, they are also used to dynamically compute the value of the **optional** slot for each child schema. When a state is predicted to be omitted, the optional slot value becomes true for that cycle of input and database response. The **optional** and **concepts-required** slots are important for determining when a goal state is complete.

3.2.1. Algorithm for Prediction Generation

These structures are used to derive predictions by the following process. When an incoming utterance and database response are processed, we select the most likely plan steps executed. If a plan step is not complete, then our most constraining prediction set reflects the assumption that the user will complete the plan step. The other prediction layers do not change. If one or more plan steps are complete, we identify the next goal states to which a system user could transit. Identifying possible next states is the basis for each layer of predictions. Next, we take all of the possible next plan step which would follow from the just completed step and store them. Following this, we apply our less reliable knowledge sources to further prune the set of next, most likely steps. To do this, we first use any and all knowledge of user ordering preferences and states which could be omitted. Then we back off first on the ordering information and then on the states which could be omitted. Then, since all goal states and possible problem solving plans are represented in a hierarchical manner, we progressively move up a layer in the hierarchy of incomplete, yet active plan steps or goal states to determine the state to which the user could next progress.[1] Once we have determined the next states, we can take the concepts associated with the states and compute restrictions on their expansions, restrictions on references given the state and the context, and restrictions on partial utterances.

Once the predictions are generated, they are expanded into potential surface forms and used by the speech recognition module to guide the pattern matching process, as described below.

4. Use of Predictions to Guide Recognition

The idea behind the MINDS system is to use pragmatic knowledge to reduce the amount of search performed by the speech recognizer thereby reducing recognition errors caused by ambiguity and word confusion. Hence, pragmatic knowledge is used predictively. These predictions take the form of semantic concepts with restrictions on their children and restrictions on methods of referencing the concepts. The underlying motivation for using a semantic representation was that speech recognizers can be guided by using a semantic grammar. Furthermore, a semantic grammar can be represented with non-terminal rewrite rules which group semantically related surface forms. Thus, once the layered predictions are generated, appropriate portions of the semantic grammar are "activated" and restrictions are placed on the expansion of rewrite rules as dictated by the predictions. The expanded "active" grammar is then used to guide the speech recognizer.

4.1. Expanding Predictions into Potential Surface Forms

To expand the prediction sets, we must relate the abstract concepts to words sequences which represent the conceptual meaning of the concepts.

For each concept, we have a partially precompiled set of possible surface forms which can be used in actual utterances. These individual concepts usually expand into noun phrases.

In addition to the individual concepts, we have a complete semantic network grammar which is indexed according to the combinations of semantic concepts expressed. The semantic network grammar is partitioned into subnets. A subnet defines allowable syntactic surface forms to express a particular combination of semantic concepts. *For example, all the ways for asking about a ship's mission are grouped into subnets*. The subnets are also partitioned along syntactic lines, such as ellipsis (a partial utterance), single anaphora (we, he), plural anaphora (they, them, those) and definite reference (the). This multidimensional indexing allows predictions about syntactic forms as well as concepts. Thus, the surface forms associated with each combination of semantic concepts are segmented into a number of subnets.

The grammar is pre-compiled into finite-state networks. The nodes of the nets represent non-terminal categories which expand into words. In this way, we can precompile our subnets for efficiency and still permit the predictions to restrict the expansion of non-terminals into words. This also allows us to add additional words to the system lexicon without modifying the grammar.

[1]The algorithm is somewhat simplified for purposes of this discussion. A forthcoming paper will discuss predictions re: clarification subdialogs and non-hierarchical open focus spaces.

4.1.1. Algorithm

As illustrated above, the grammar is multidimensionally segmented into subnets. Our algorithm for using this information to translate each set of layered predictions into a form usable by the speech recognition module is as follows.

First we find the set of subnets which contain one or more of the predicted semantic concepts. Forms that violate predictions on ellipsis or anaphora are pruned from this set. This set defines the nets to be active for the next utterance. Once the set of subnets is defined, we look for all the semantic concept categories, and check if their membership has been reduced by the predictions from the dialog module. This step represents a restriction on concept words that are active. The module then forms an active lexicon list and grammar based on the resulting subnets and restrictions derived from this algorithm.

The final expansion of predictions brings together the partitioned semantic networks that are currently predicted and the concepts in their surface forms. Through an extensive set of indexing, we intersect all predicted concept expressions with all the predicted semantic networks. This operation dynamically generates one combined semantic network grammar which embodies all the dialog level and sentence level constraints on the sentences which can be matched.

This operation is repeated for each set of predictions and results in a set of layered semantic networks. These networks are used by the recognizer to guide the pattern matching process.

To illustrate this point, let us assume that the frigate "Spark" has a disabled sps-48 radar. One layer of our predictions expects the user to ask when it will be repaired. The dialog tracking module predicts the "shipname" concept restricted to the value "Spark", the estimated time of repair concept and the "ship-capabilities" concept, restricted to radar and SPS-48. Single anaphoric reference to the ship is also expected, but ellipsis is not meaningful at this point. The current damage assessment dialog phase allows queries about features of a single ship.

During the expansion of the concepts, we find the word nets such as "the ship", "this ship", "the ship's", "this ship's", "it", "its", "Spark" and "Spark's". We also find the word nets for the radar capabilities such as "surface search radar", "sps-48", "radar", etc., and word nets for repair questions.

We then intersect these with the sentential forms allowed during this dialog phase. Thus we obtain the nets for phrases like "Display/list etr/estimated time to repair/estimated repair time/projected time for repair on/for surface search radar/sps-48/radar/sps-48 surface search radar", and "Display/what is/ its/Spark's/this ship's/the ship's etr/projected repair time/", and many more. This semantic network now represents a maximally constrained grammar which reflects the constraints embodied in this layer of predictions.

4.2. Recognizing Speech Using Dynamic Networks

As explained above, predictions are used to define an active set of subnets and an active set of words to be used in processing the next utterance. We use the Sphinx system as the basis for our recognizer. It has been modified to use finite state nets to control word transitions instead as opposed to word-pairs or bigrams. Sphinx creates word models by concatenating Hidden Markov Models of phonemes. These word networks are precompiled.

During recognition, the speech module performs a time-syncronous beam search. The search traces through the active nodes of our nets to control word transitions. As the search exits a word it forms a set of words to transit to from successor states in the nets. Only the active finite state nets and active words are used to compute the successor word set. The search then transits to the words in this set. Paths falling below a threshold score are pruned. The network is used to allow only "legal" transitions. It does not affect the score of a path but simply restricts words which can continue the path.

The recognizer is given several sets of predictions which are successively more general (less constraining). The most constraining set is used first. If no string is found which exceeds a threshold score, the input is reprocessed using the next more general set of predictions. If an acceptable recognition is not found using the most general set of predictions, the entire set of nets is used.

After the input has been processed, the word string with the best score is passed back to the system for parsing. In addition to the word string, the subnet matched, the overall score and individual word scores are passed back.

5. Results

The above described use of plans in speech recognition is currently embodied in the MINDS system (Young and Ward, 1988; Young, Hauptmann and Ward, 1988; Hauptmann and Young, 1988). MINDS is a multimedia interactive dialog system where users solve problems by interacting with a database. Users can speak, type or point to input information and both the system and the user can initiate clarification dialogs when appropriate. It uses an adapted version of the SPHINX (Lee, 1988) speech recognition system with a 1000 word vocabulary. Its task domain is naval resource management. Here users must query a relational database to determine whether a disabled vessel should be replaced with another vessel, scheduled for a later repair, or whether the mission should be delayed.

To test the ability of our layered predictions to both reduce search space and to improve speech recognition performance, we performed two experiments. The first experiment assessed perplexity reduction enabled by the predictive use of pragmatic knowledge. The second experiment measured improvement in recognition accuracy rates resulting from the use of layered predictions. Both studies used an independent test set. This means that the utterances processed by the system to obtain the experimental results had not been previously seen by the system. Furthermore, the test set did not include any clarificiation dialogs.

5.1. Test and Training Sets

Our test data (10 scenarios) were adapted versions of three problem solving sessions taken from the TONE database. The TONE database is a set of transcripts from Naval personnel solving problems about what to do with a disabled vessel. The personnel must determine whether to delay a mission, find a replacement vessel or schedule a repair for a later date. They use a database to find necessary problem solving information. In addition to the three scenarios from the TONE database, we created seven additional sessions by paraphrasing the original three. These scenarios were not used to train upon.

Our training data were five different problem solving scenarios from the TONE database. The training scenarios were used for writing grammars and developing user models. Problem solving plans were derived from an abstract description of the stages and options available to a problem solver. The abstract plan descriptions were provided by the Navy.

Our database was different from the one used in gathering the TONE transcripts. While it contained the same fields, the information about particular ships differed across the two databases. To enable testing with the TONE transcripts, we had to adapt the test scenarios. Our adaptations consisted of the following:

- Shipnames were changed to correspond to those in our database.
- Lexical entries not in our lexicon (such as 'employment schedule') were replaced with equivalent concepts from our lexicon (such as 'mission' and 'mission importance').
- Database inconsistencies were resolved in favor of the CMU database. For example, if in the naval database, ship X required capability Y for its mission but in the CMU database ship, X required mission capability Z, all references in a scenario to Y were replaced with references to Z.

These adaptations have minimal impact on the integrity of the data.

5.2. Reduction in Search Space

Our first experiment was designed to test the search space reduction resulting from applying pragmatic constraints. Thus, we used all 10 of our test scenarios. The scenarios contained an average of 9 sentences.

To measure the constraint imposed by the knowledge sources, we use an index called perplexity. This is an information theoretic measure that is widely used in speech systems to characterize the constraint provided by a grammar. Perplexity represents the geometric mean of the number of alternative words at any point. Search space size for a given test sentence is computed by raising perplexity to the number of words in the sentence.

To measure the reduction in perplexity and search space it was necessary to collect test set perplexity measurements for each of the parsed sentences in two conditions:

- Total domain grammar alone
- Using predictions

Test set perplexity is the perplexity of the actual sentence parsed. It is different than total grammar perplexity because it takes into account only those alternatives which are legal next words given the grammar.

To measure the test set perplexity of all the sentences in each of the test scenarios using the entire system grammar is relatively straight forward. However, measuring the test set perplexity of sentences which are parsed with layered predictions is not. Since prediction layers fail, we must report the perplexity of the layers which were successful. However, since some layers are non-overlapping, the number we report is the perplexity of the successful prediction layer merged with all the unsuccessful layers attempted.

As seen in Table 1, test set perplexity was reduced in excess of an order of magnitude, from 279.2 to 17.8.

Reduction in Branching Factor and Search Space		
Constraints used:	grammar	layered predictions
Test Set Perplexity	279.2	17.8
Search Space	3.81×10^{19}	1.01×10^{9}

Put differently, the knowledge sources reduced the search space for lexical entries by 9 orders of magnitude on the average 8 word sentence when the predictions were expanded into potential surface expression forms for future utterances.

5.3. Recognition Performance

To evaluate the effects of using layered predictions on recognition performance we used 10 speakers (8 male, 2 female) who had not been used to train the recognizer. Each speaker read 20 sentences from the adapted test set provided by the Navy. Each of these utterances was recorded. The speech recordings were then run through the SPHINX recognition system in two conditions:

- using the system grammar (all legal sentences)
- using the successful prediction layer merged with all unsuccessful layers

The results can be seen in Table 2.

As can be seen, the system performed significantly better with the predictions. Error rate decreased by a factor of five. Perhaps more important, however, is the nature of the errors. In the "with predictions" condition, 89 percent of the insertions and deletions were the word "the". Additionally, 67 percent of the substitutions were "his" for "its". Furthermore, none of the errors in the "with predictions" condition resulted in an incorrect database query. Hence, semantic

Recognition Performance		
Constraints used:	grammar	layered predictions
Test Set Perplexity	242.4	18.3
Word Accuracy	82.1	96.5
Semantic Accuracy	85%	100%
Insertions	0.0%	0.5%
Deletions	8.5%	1.6%
Substitutions	9.4%	1.4%

accuracy was 100%.

6. Summary

In summary, by identifying and using knowledge sources which can intelligently reduce search space, we progress toward developing robust, interactive problem solving environments where speech is the primary mode of communication. One such knowledge source is pragmatics. The use of layered predictions derived from pragmatic knowledge sources appears to be a powerful technique for improving speech recognition and reducing search space. Layered predictions allow the recognition system to capitialize upon pragmatic knowledge sources without impairing the system's ability to recognize less likely utterances. The more consistent the users behavior, the better the recognition. As user behavior deviates, recognition accuracy degrades gracefully and the system is capable of recovering and generating further pragmatic predictions based upon both the users expected and less expected behavior. However, as domains continue to scale up and we begin to process spontaneously generated speech, additional knowledge sources will become increasingly important.

References

1. Allen, J. F. and Perrault, C. R. "Analyzing Intention in Utterances". *Artificial Intelligence 15*, 3 (1980), 143-178.

2. Biermann, A., Rodman R., Ballard B., Betancourt, T., Bilbro, G., Deas, H., Fineman, L., Fink, P., Gilbert, K., Gregory, D. and Heidlage, F. Interactive natural language problem solving: A pragmatic approach. Conference on Applied Natural Language Processing, 1983, pp. 180 - 191.

3. Borghesi, L. and Favareto, C. Flexible Parsing of Discretely Uttered Sentences. COLING-82, Association for Computational Linguistics, Prague, July, 1982, pp. 37 - 48.

4. Carbonell, J. G. "POLITICS: Automated Ideological Reasoning.". *Cognitive Science 2*, 1 (1978), 27-51.

5. Cohen, P. R. and Perrault, C. R. "Elements of a Plan-Based Theory of Speech Acts". *Cognitive Science 3* (1979), 177-212.

6. Erman, L.D. and Lesser, V.R. The Hearsay-II Speech Understanding System: A Tutorial. In Lea, W.A., Ed., *Trends in Speech Recognition*, Prentice-Hall, Englewood Cliffs, NJ, 1980, pp. 340 - 360.

7. Fikes, R. E. and Nilsson, N. J. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving". *Artificial Intelligence 2* (1971), 189-208.

8. Fink, P. K. and Biermann, A. W. "The Correction of Ill-Formed Input Using History-Based Expectation With Applications to Speech Understanding". *Computational Linguistics 12* (1986), 13-36.

9. Grosz, B. J. and Sidner, C. L. "Attention, Intentions and the Structure of Discourse". *Computation Linguistics 12* (1986), 175-204.

10. Hauptmann, A. G., Young, S. R. and Ward, W. H. Using Dialog Level Knowledge Sources to Improve Speech Recognition. Proceedings of the Seventh National Conference on Artificial Intelligence, 1988.

11. Kimball, O., Price, P., Roucos, S., Schwartz, R., Kubala, F., Chow, Y.-L., Haas, A., Krasner, M. and Makhoul, J. Recognition Performance and Grammatical Constraints. Proceedings of the DARPA Speech Recognition Workshop, Science Applications International Corporation Report Number SAIC-86/1546, 1986, pp. 53 - 59.

12. Lea, W.A. (Ed.). *Trends in Speech Recognition*. Prentice-Hall, Englewood Cliffs, NJ, 1980.

13. Lee, K. *SPHINX: Large Vocabulary, Speaker-Independent Speech Recognition*. Ph.D. Th., Carnegie-Mellon University, 1988.

14. Levinson, S. E. and Shipley, K. L. "A Conversational-Mode Airline Information and Reservation System Using Speech Input and Output". *The Bell Systems Technical Journal 59* (1980), 119 - 137.

15. Levinson, S. E. and Rabiner, L. R. "A Task-Oriented Conversational Mode Speech Understanding System". *Bibliotheca Phonetica 12* (1985), 149-196.

16. Litman, D. J. and Allen, J. F. "A Plan Recognition Model for Subdialogues in Conversation". *Cognitive Science 11* (1987), 163-200.

17. Lowerre, B. and Reddy, R. The Harpy Speech Understanding System. In Lea, W.A., Ed., *Trends in Speech Recognition*, Prentice-Hall, Englewood Cliffs, NJ, 1980, pp. 340 - 360.

18. Newell, A. and Simon, H. A.. *Human Problem Solving*. New Jersey: Prentice-Hall, 1972.

19. Sacerdoti, E. D. "Planning in a Hierarchy of Abstraction Spaces". *Artificial Intelligence 5*, 2 (1974), 115-135.

20. Wilensky, R. *Understanding Goal-Based Stories*. Ph.D. Th., Yale University, Sept. 1978.

21. Wilensky, R.. *Planning and Understanding*. Addison Wesley, Reading, MA, 1983.

22. Young, S. R., Hauptmann, A. G. and Ward, W. H. An Integrated Speech and Natural Language Dialog System: Using Dialog Knowledge in Speech Recognition. Tech. Rept. CMU-CS-88-128, Carnegie Mellon University Computer Science Technical Report, 1988. also submitted.