EVALUATION OF THE CMU ATIS SYSTEM

Wayne Ward

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pa 15213

ABSTRACT

The CMU Phoenix system is an experiment in understanding spontaneous speech. It has been implemented for the Air Travel Information Service task. In this task, casual users are asked to obtain information from a database of air travel information. Users are not given a vocabulary, grammar or set of sentences to read. They compose queries themselves in a spontaneous manner. This task presents speech recognizers with many new problems compared to the Resource Management task. Not only is the speech not fluent, but the vocabulary and grammar are open. Also, the task is not just to produce a transcription, but to produce an action, retrieve data from the database. Taking such actions requires parsing and "understanding" the utterance. Word error rate is not as important as utterance understanding rate.

Phoenix attempts to deal with phenomena that occur in spontaneous speech. Unknown words, restarts, repeats, and poorly formed or unusual grammar are common is spontaneous speech and are very disruptive to standard recognizers. These events lead to misrecognitions which often cause a total parse failure. Our strategy is to apply grammatical constraints at the phrase level and to use semantic rather than lexical grammars. Semantics provide more constraint than parts of speech and must ultimately be delt with in order to take actions. Applying constraints at the phrase level is more flexible than recognizing sentences as a whole while providing much more constraint than word-spotting. Restarts and repeats are most often between phase occurences, so individual phrases can still be recognized correctly. Poorly constructed grammar often consists of well-formed phrases, and is often semantically well-formed. It is only syntactically incorrect. We associate phrases by frame-based semantics. Phrases represent word strings that can fill slots in frames. The slots represent information which the frame is able to act on.

The current Phoenix system uses a bigram language model with the Sphinx speech recognition system. The top-scoring word string is passed to a flexible frame-based parser. The parser assigns phrases (word strings) from the input to slots in frames. The slots represent information content needed for the frame. A beam of frame hypotheses is produced and the best scoring one is used to produce an SQL query.

INTRODUCTION

Understanding spontaneous speech presents several problems not found in transcribing read speech input. Spontaneous speech is often not fluent. It contains stutters, filled pauses, restarts, repeats, interjections, etc. Casual users do not know the lexicon and grammar used by the system. It is therefore very difficult for a speech understanding system to achieve good coverage of the lexicon and grammar that subjects might use. Also, the task of the system is not just to produce a transcription, but to produce an action. Taking such actions requires parsing and "understanding" the utterance. Word error rate is not as important as utterance understanding rate.

The Air Travel Information Service task is being used by several DARPA-funded sites to develop and evaluate speech understanding systems for database query tasks. In the ATIS task, novice users are asked to perform a task that requires getting information from the Air Travel database. This database contains information about flights and their fares, airports, aircraft, etc. The only input to the system is by voice. Users compose the questions themselves, and are allowed to phrase the queries any way they choose. No explicit grammar or lexicon is given to the subject.

At Carnegie Mellon University, we have been developing a system, called Phoenix, to understand spontaneous speech [1] [2] [3]. We have implemented an initial version of this system for the ATIS task. This paper presents the design of the Phoenix system and its current status. We also report system evaluation results for the DARPA Feb91 test.

THE PHOENIX SYSTEM

Some problems posed by spontaneous speech are:

- User noise - breath noise, filled pauses and other user generated noise
- Environment noise - door slams, phone rings, etc.
- Out-of-vocabulary words - The subject says words that the system doesn't know.
- Grammatical coverage - Subjects often use grammatically ill-formed utterances and restart and repeat phrases.

Phoenix address these problems by using non-verbal sound models, an out-of-vocabulary word model and flexible parsing.

Non-Verbal Sound Models

Models for sounds other than speech have been shown to significantly increase performance of HMM-based recognizers for noisy input. [2] [4] In this technique, additional models are added to the system that represent non-verbal sounds, just as word models represent verbal sounds. These models are trained exactly as if they were word models, but using the noisy input. Thus, sounds that are not words are allowed to map onto tokens that are also not words.

Out-of-vocabulary Word Model

This module has not yet been implemented. In order to deal with out-of-vocabulary words, we will use a technique essentially like the one presented by BBN. [5] We will create an explicit model for out-of-vocabulary words. This model allows any triphone (context dependent phone) to follow any other triphone (given of course that the context is the same) with a bigram

probability model. The bigrams are to be trained from a large dictionary of English pronunciations.

Flexible Parsing

Our concept of flexible parsing combines frame based semantics with a semantic phrase grammar. We use a frame based parser similar to the DYPAR parser used by Carbonell, et al. to process ill-formed text, [6] and the MINDS system previously developed at CMU. [7] Semantic information is represented in a set of frames. Each frame contains a set of slots representing pieces of information. In order to fill the slots in the frames, we use a partitioned semantic phrase grammar. Each slot type is represented by a separate finite-state network which specifies all ways of saying the meaning represented by the slot. The grammar is a semantic grammar, non-terminals are semantic concepts instead of parts of speech. The grammar is also written so that phrases can stand alone (be recognized by a net) as well as being embedded in a sentence. Strings of phrases which do not form a grammatical English sentence are still parsed by the system. The grammar is compiled into a set of finite-state networks. It is partitioned in the sense that, instead of one big network, there are many small networks. Networks can "call" other networks, thereby significantly reducing the overall size of the system. These networks are used to perform pattern matches against input word strings. This general approach has been described in earlier papers. [1] [3]

The operation of the parser can be viewed as "phrase spotting". A beam of possible interpretations are pursued simultaneously. An interpretation is a frame with some of its slots filled. The finite-state networks perform pattern matches against the input string. When a phrase is recognized, it attempts to extend all current interpretations. That is, it is assigned to slots in active interpretations that it can fill. Phrases assigned to slots in the same interpretation are not allowed to overlap. In case of overlap, multiple interpretations are produced. When two interpretations for the same frame end with the same phrase, the lower scoring one is pruned. This amounts to dynamic programming on series of phrases. The score for an interpretation is the number of input words that it accounts for. At the end of the utterance, the best scoring interpretation is output.

In our system, slots (pattern specifications) can be at different levels in a hierarchy. Higher level slots can contain the information specified in several lower level slots. These higher level forms allow more specific relations between the lower level slots to be specified. In the utterance "leaving denver and arriving in boston after five pm", "leaving denver" is a [depart_loc] and "arriving in boston" is an [arrive_loc], but there is ambiguity as to whether "after 5 pm" is [depart_time_range] or [arrive_time_range]. The existence of the higher level slot [ARRIVE] allows this to be resolved. One rewrite for the slot [ARRIVE] is ([arrive_loc] [arrive_time_range]) in which the two lower level slots are specfically associated. Thus two interpretations for this utterance are produced,

```
leaving denver and arriving
in boston after 5 pm

1
[depart_loc]  leaving denver
[arrive_loc]  arriving in boston
[depart_time_range]  after 5 pm

2
[depart_loc]  leaving denver
[ARRIVE]
   [arrive_loc]  arriving in boston
   [arrive_time_range]  after 5 pm
```

In picking which interpretation is correct, higher level slots are preferred to lower level ones because the associations between concepts is more tightly bound, thus the second (correct) interpretation is picked here.

Our strategy is to apply grammatical constraints at the phrase level and to associate phrases in frames. Phrases represent word strings that can fill slots in frames. The slots represent information which, taken together, the frame is able to act on. We also use semantic rather than lexical grammars. Semantics provide more constraint than parts of speech and must ultimately be delt with in order to take actions. Applying constraints at the phrase level is more flexible than recognizing sentences as a whole while providing much more constraint than word-spotting. Restarts and repeats are most often between phases, so individual phrases can still be recognized correctly. Poorly constructed grammar often consists of well-formed phrases, and is often semantically well-formed. It is only syntactically incorrect.

System Structure

The overall structure of our current system is shown in Figure 1. We use the Sphinx system as our recognizer module [8]. Sphinx is a speaker independent continuous speech recognition system.

Currently the recognizer and parser are not integrated. The speech input is digitized and vector quantized and then passed to the Sphinx recognizer. The recognizer uses a bigram language model to produce a single best word string from the speech input. This word string is then passed to the frame-based parser which assigns word strings to slots in frames as explained above.

The slots in the best scoring frame are then used to build objects. In this process, all dates, times, names, etc. are mapped into a standard form for the routines that build the database query. The objects represent the information that was extracted from the utterance. There is also a currently active set of objects which represent constraints from previous utterances. The new objects created from the frame are merged with the current set of objects. At this step ellipsis and anaphora are resolved. Resolution of ellipsis and anaphora is relatively simple in this system. The slots in frames are semantic, thus we know the type of object needed for the resolution. For ellipsis, we add the new objects. For anaphora, we simply have to check that an object of that type already exists.

Each frame has an associated function. After the information is

Input	% True	% False	% No Answer	Weighted Score
Transcript	80.7	16.6	2.8	64.0
Speech	61.4	26.9	11.7	34.5

Table 1: Phoenix results for Feb91 Class-A test set

Src	Subs	Del	Ins	Error
Word	19.3	6.8	2.6	28.7
String	79.1	-	-	79.1

Table 2: Recognition error rates for Class-A

extracted and objects built, the frame function is executed. This function takes the action appropriate for the frame. It builds a database query (if appropriate) from objects, sends it to SYBASE (the DataBase Management System we use) and displays output to the user.

RESULTS

Our current system has a lexicon of 710 words and uses a bigram language model of perplexity 49. Six noise models are included in the lexicon. We used the version of Sphinx produced by Hon [9], which includes between-word triphone models. The vocabulary-independent phone models generated by Hon [9] were used to compile the word models for the system. No task specific acoustic training was done. We have not yet added the out-of-vocabulary models to the system.

The DARPA ATIS0 training set consists of approximately 700 utterances gathered by Texas Instruments and distributed by NIST. This data was gathered and distributed before the June 1990 evaluations. The data was gathered using a "wizard" paradigm. Subjects were asked to perform an ATIS scenario. They were given a task to perform and told that they were to use a speech understanding computer to get information. A hidden experimenter listened to the subjects and provided the appropriate information from the database. The transcripts from this set were used to train our language model. This includes the bigram model for the recognizer and the grammar for the parser. Since this amount of data is not nearly enough to train a language model, we chose to "pad" our bigrams. Bigrams were generated based on tag pairs rather than word pairs. Words in our lexicon were put into categories represented by tags. The June90 training corpus was tagged according to this mapping. We then generated a word-pair file from the Phoenix finite-state ATIS grammar. This file was used to initialize the tag bigram counts. The tagged corpus was then used to add to the counts and the bigram file was generated. It is a "padded" bigram in the sense that the grammar is used to insure a count of at least 1 for all "legal" tag pairs. This procedure yielded a bigram language model which has perplexity 39 for the ATIS0 test set.

The DARPA ATIS1 test (for the February 1991 evaluations) has two mandatory test sets, the class A set and the class D1 set.

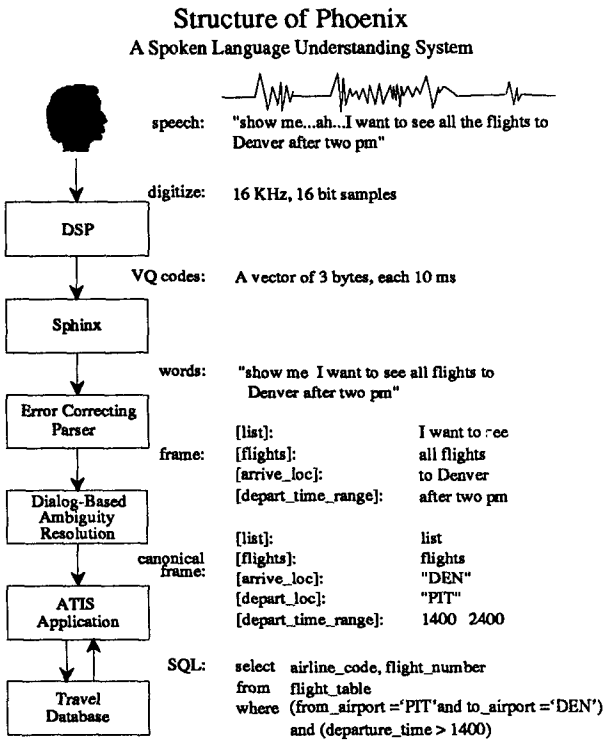

Figure 1: Structure of the Phoenix system

The class A set contains 145 utterances that are processed individually without context. All utterances in the test set were "Class-A", that is, answerable, context independent and with no disfluencies. The class D1 set contains 38 utterance pairs. These are intended to test dialog capability. The first utterance of a pair is a Class-A utterance that sets the context for the second. Only scores for the second utterance are reported for this set.

We processed both transcript and speech input for each set. Tables 1-4 show the results of this evaluation.

Utterances were scored correct if the answer output by the

Input	True	False	No Answer	Weighted Score
Transcript	60.5	34.2	5.2	26.3
Speech	39.4	55.2	5.2	-15.8

Table 3: Phoenix results for Feb91 Class-D1 test set

Src	Subs	Del	Ins	Error
Word	17.6	8.6	0.7	26.9
String	77.6	-	-	77.6

Table 4: Recognition error rates for Class-D1

Source	% of Total Errors
Grammatical Coverage	25
Semantic Coverage	20
Wrong CAS Field	25
Unanswerable	10
Application Coding Errors	20

Table 5: Analysis of Errors for Class-A NL

system matched the reference answer for the utterance. The reference answer is database output, not a word string. Systems are allowed to output a NO_ANSWER response, indicating that the utterance was misunderstood. Any output that was not correct or NO_ANSWER was scored incorrect. The Weighted Score is computed as (1- (2*percent_false + percent_NO_ANSWER)).

Table 1 shows the results for class A utterances. For these, the system produced the correct answer for 80.7 percent of the transcript input and 61.4 percent of the speech input. The performance for transcript input reflects the grammatical and semantic coverage of the parser and application program. The performance for the speech input reflects additional errors made in the recognition stage. Recognition performance for these utterances is shown in Table 2. Word substitutions, deletions and insertions are summed to give the word error measure of 28.7 percent. A string error rate of 79 percent means that only twenty one percent of the utterances contained no errors. However, 61 percent of the utterances gave correct answers. This illustrates the ability of the parser to handle minor misrecognitions in the recognized string.

The D1 test set is designed to provide a test of dialog capability. The utterances are specified in pairs. The first utterance is processed normally and is used to set the context for the second utterance of the pair. Missing the first utterance can lead to incorrectly interpreting the second. Tables 3 and 4 show the understanding performance and speech recognition rates for the D1 test set. While the recognition results are comparable to those for set A, the understanding performance is significantly worse. This is due in large part to utterances in which we missed the first utterance, causing the context for the second to be wrong.

We feel that recognition error rates for spontaneous input will improve considerably with the addition of out-of-vocabulary models and with better lexical and grammatical coverage.

ERROR ANALYSIS

In order to interpret the performance of the system, it is useful to look at the source of the errors. Table 5 shows the percentage of errors from various sources.

Twenty five percent of our errors were a result of lack of grammatical coverage. This includes unknown words for concepts that the system has. For example, the system knew day names (Monday, Tuesday, etc) but not plural day names (Mondays, etc) since these had not been seen in the training data. This category also contains errors where all words were known but the specific word sequence used did not match any phrase patterns.

Twenty percent of the errors were due to a lack of semantic coverage. In this case, there were no frames for the type of question being asked or no slots for the type of information being provided. For example, one utterance requested "a general description of the aircraft". Our system allows you to ask about specific attributes of an aircraft but does not have the notion of "general description" which maps to a subset of these attributes.

Twenty five percent of the errors were due to outputting the wrong field from the database for the CAS answer. In these cases, the utterance was correctly understood and a reasonable answer was output, but it was not the specific answer required by the CAS specifications. For example, when asked for cities near the Denver airport, we output the city name "DENVER" rather than the city code "DDEN" as required by CAS.

Ten percent of the errors were due to utterances that our system considered unanswerable. For CAS evaluation runs, we map all system error messages to a NO_ANSWER response. For example, one utterance asked for ground transportation from Atlanta to Baltimore. Our system recognized that this was outside the abilities of the database and generated an error message that was mapped to NO_ANSWER. The reference answer was the null list "()".

The other twenty percent of the errors were due to coding bugs in the back end.

The first two categories (grammatical and semantic errors) are errors in the "understanding" part of the system. Forty five percent of our total errors were due to not correctly interpreting the input. The other fifty five percent of the errors were generation errors. That is, the utterance was correctly interpreted but the correct answer was not generated.

FUTURE PLANS

Our next step in the evolution of the Phoenix system will be to integrate the recognition and parsing. We will use the pattern matching networks to drive the word transitions in the recognition search rather than a bigram grammar.

ACKNOWLEDGMENTS

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 5167, under contract number N00039-85-C-0163. 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 US Government.

REFERENCES

1. Ward, W., "Understanding Spontaneous Speech", *Proceedings of the DARPA Speech and Natural Language Workshop*, 1989, pp. 137, 141.
2. Ward, W., "Modelling Non-verbal Sounds for Speech Recognition", *Proceedings of the DARPA Speech and Natural Language Workshop*, 1989, pp. 47, 50.
3. Ward, W., "The CMU Air Travel Information Service: Understanding Spontaneous Speech", *Proceedings of the DARPA Speech and Natural Language Workshop*, 1990.
4. Wilpon, J.G., Rabiner, L.R., Lee, C.H., Goldman, E.R., "Automatic Recognition of Vocabulary Word Sets in Unconstrained Speech Using Hidden Markov Models", *in press Transactions ASSP*, 1990.
5. Asadi, A., Schwartz, R., Makhoul, J., "Automatic Detection Of New Words In A Large Vocabulary Continuous Speech Recognition System", *Proceedings of the DARPA Speech and Natural Language Workshop*, 1989, pp. 263, 265.
6. Carbonell, J.G. and Hayes, P.J., "Recovery Strategies for Parsing Extragrammatical Language", Tech. report CMU-CS-84-107, Carnegie-Mellon University Computer Science Technical Report, 1984.
7. Young, S. R., Hauptmann, A. G., Ward, W. H., Smith, E. T. and Werner, P., "High Level Knowledge Sources in Usable Speech Recognition Systems", *Communications of the ACM*, Vol. 32, No. 2, 1989, pp. 183-194.
8. Lee, K.-F., *Automatic Speech Recognition: The Development of the SPHINX System*, Kluwer Academic Publishers, Boston, 1989.
9. Hon, H.W., Lee, K.F., Weide, R., "Towards Speech Recognition Without Vocabulary-Specific Training", *Proceedings of the DARPA Speech and Natural Language Workshop*, 1989, pp. 271, 275.