RECENT IMPROVEMENTS IN THE CMU SPOKEN LANGUAGE UNDERSTANDING SYSTEM

Wayne Ward and Sunil Issar

School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15217

ABSTRACT

We have been developing a spoken language system to recognize and understand spontaneous speech. It is difficult for such systems to achieve good coverage of the lexicon and grammar that subjects might use because spontaneous speech often contains disfluencies and ungrammatical constructions. Our goal is to respond appropriately to input, even though coverage is not complete. The natural language component of our system is oriented toward the extraction of information relevant to a task, and seeks to directly optimize the correctness of the extracted information (and therefore the system response). We use a flexible frame-based parser, which parses as much of the input as possible. This approach leads both to high accuracy and robustness. We have implemented a version of this system for the Air Travel Information Service (ATIS) task, which is being used by several ARPA-funded sites to develop and evaluate speech understanding systems. Users are asked to perform a task that requires getting information from an Air Travel database. In this paper, we describe recent improvements in our system resulting from our efforts to improve the coverage given a limited amount of training data. These improvements address a number of problems including generating an adequate lexicon and grammar for the recognizer, generating and generalizing an appropriate grammar for the parser, and dealing with ambiguous parses.

1. INTRODUCTION

Understanding spontaneous speech presents several problems that are not found either in recognizing read speech or in parsing written text. Since the users are not familiar with the lexicon and grammar used by the system, it is very difficult for a speech understanding system to achieve good coverage of the lexicon and grammar that subjects might use. Spontaneous speech often contains ungrammatical constructions, stutters, filled pauses, restarts, repeats, interjections, etc. This causes problems both for the recognizer and the parser.

Stochastic language models tend to produce more robust recognition than grammar based models. They can be smoothed to allow for unseen word sequences and their scope is short enough to "get back on track" after an error. The parsing and understanding component also must be robust to the phenomena in spontaneous speech and to recognition errors. Even though the speech is disfluent and gramatically ill-formed, the relevant information is still consistent most of the time. We therefore try to model the information in an utterance rather than its grammatical structure. The natural language component of our system is oriented toward the extraction of information relevant to a task, and seeks to directly optimize the correctness of the extracted information (and therefore the system response). We use a flexible frame-based parser, which parses as much of the input as possible. This approach leads both to high accuracy and robustness.

We have implemented a version of this system for the ARPA Air Travel Information Service (ATIS) task. Users are asked to perform a task that requires getting information from an Air Travel database. They must interact with the system by voice to find a solution. In this paper, we describe recent improvements in our system resulting from our efforts to increase the coverage given a limited amount of training data. These improvements address a number of problems mentioned above:

- Generating and generalizing an appropriate grammar for the parser
- Generating a lexicon and language model for the recognizer
- Resolving ambiguous parses with context

In addition, we improved the basic performance of the parser and added a rejection mechanism.

2. SYSTEM OVERVIEW

The CMU spoken language understanding system is called Phoenix, and has been described in previous papers [4, 3]. It is neccessary here to give a brief description of the system in order to understand the context within which we were making changes.

Our system has a loose coupling between the recognition and parsing stages. The recognizer uses stochastic language models to produce a single word string hypothesis. This hypothesis is then passed to a parsing module which uses semantic grammars to produce a semantic representation for the input utterance. We use a stochastic language model in recognition because of its robustness and efficiency. The parsing stage must then be tolerant of mistakes due to disfluencies and misrecognitions.

The parser outputs a frame with slots filled from the current utterance. Information from the current frame is integrated with information from previous frames to form the current context. A set of tests is applied to determine whether to reset context. For example if new depart and arrive locations are specified, old context is cleared. The current context is then used to build an SQL query.

2.1. Recognition

The CMU Sphinx-II system [2] uses semi-continuous Hidden Markov Models to model context-dependent phones (triphones), including between-word context. The phone models are based on senones, that is, observation distributions are shared between corresponding states in similar models. The system uses four codebooks: Mel-scale cepstra, 1st and 2nd difference cepstra, and power. An observation probability is computed using a mixture of the top 4 distributions from each codebook.

The recognizer processes an utterance in four steps.

1. It makes a forward time-synchronous pass using full between-word models, Viterbi scoring and a bigram language model. This produces a word lattice where words have one begin time but several end times.
2. It then makes a backward pass which uses the end times from the words in the first pass and produces a second lattice which contains multiple begin times for words.
3. An A* algorithm is used to generate the set of N-best hypotheses for the utterance from these two lattices. An N of 100 was used for these tests.
4. The set of N-best hypotheses is then rescored using a trigram language model. The best scoring hypothesis after rescoring is output as the result.

2.2. Parsing

Our NL understanding system (Phoenix) is designed for robust information extraction. It uses a simple frame mechanism to represent task semantics. Frames are associated with the various types of actions that can be taken by the system. Slots in a frame represent the various pieces of information relevant to the action that may be specified by the subject. For example, the most frequently used frame in ATIS is the one corresponding to a request to display some type of flight information. Slots in the frame specify what information is to be displayed (flights, fares, times, airlines, etc), how it is to be tabulated (a list, a count, etc) and the constraints that are to be used (date ranges, time ranges, price ranges, etc).

The Phoenix system uses Recursive Transition Networks to encode semantic grammars. The grammars specify word patterns (sequences of words) which correspond to semantic tokens understood by the system. A subset of tokens are considered as top-level tokens, which means they can be recognized independently of surrounding context. Nets call other nets to produce a semantic parse tree. The top-level tokens appear as slots in the frame structures. The frames serve to associate a set of semantic tokens with a function. Information is often represented redundantly in different nets. Some nets represent more complex bindings between tokens, while others represent simple stand-alone values. There is not one large sentential-level grammar, but separate grammars for each slot (there are approximately 70 of these in our ATIS system). The parse is flexible at the slot level in that it allows slots to be filled independent of order. It is not necessary to represent all different orders in which the slot patterns could occur.

Our semantic grammars are written to allow flexibility in the pattern match. The patterns for a semantic token consist of mandatory words or tokens which are necessary to the meaning of the token and optional elements. The patterns are also written to overgenerate in ways that do not change the semantics. This overgeneration not only makes the pattern matches more flexible but also serves to make the networks smaller.

The parser operates by matching the word patterns for slots against the input text. A set of possible interpretations are pursued simultaneously. The system is implemented as a top-down Recursive Transition Network chart parser for slots. As slot fillers (semantic phrases) are recognized, they are added to frames to which they apply. The algorithm is basically a dynamic programming beam search on frames. Many different frames, and several different versions of a frame, are pursued simultaneously. The score for each frame hypothesis is the number of words that it accounts for. A file of words not to be counted in the score is included. At the end of an utterance the parser picks the best scoring frame as the result. There is a heuristic procedure for resolving ties. The output of the parser is the frame name and the parse trees for its filled slots.

3. LANGUAGE MODEL GENERATION

Our system uses two different types of language models, a bigram for speech recognition and and semantic grammar for parsing [4, 3].

3.1. Parsing Grammar

The frame structures and patterns for the Recursive Transition Networks were developed by processing transcripts of subjects performing scenarios of the ATIS task. The data, which consists of around 20000 utterances from the ATIS2 and ATIS3 corpora, were gathered by several sites. A subset of this data (around 10000 utterances) has been annotated with reference answers. The details of data collection and annotation are described by Dahl [1].

The goal of our system is to extract all relevant information from the utterance. We do not attempt to parse each and every word in the utterance. However, we have a problem with the grammar coverage if we miss one of the content words in the utterance. Let us look at some of the sentences from the December 1993 ARPA evaluation where grammar coverage was a problem:

(x0s032sx) list *AIRPORT -DESIGNATIONS for flights from st. petersburg

(8k8012sx) find a flight round trip from los angeles to tacoma washington with a stopover in san francisco *NOT -EXCEEDING the price of three hundred dollars for june tenth nineteen ninety three

(g02084sx) list *THE -ORIGINS for alaska airlines

(g0d014sx) *ARE -SNACKS *SERVED on tower air

(i0k05esx) list *THE -DISTANCES to downtown

(i0k0eesx) list *THOSE -DISTANCES from los angeles to downtown

In these sentences, the words preceded by '-' did not occur in the grammar. However, we could parse the following sentences which are similar to these sentences. The differences are highlighted in bold letters.

list airport **designation** for flights from st. petersburg

find a flight round trip from los angeles to tacoma washington with a stopover in san francisco not **more than** the price of three hundred dollars for june tenth nineteen ninety three

list the **origin** for alaska airlines

are **snack** served on tower air

list the **distance** to downtown

list those **distance** from los angeles to downtown

Some of the words that occurred in the test set, but were not covered by our grammar are as follows:

ALONG COMBINATION DESIGNATIONS
DISTANCES EVER EXCEEDING INFORMATIONS
ORIGINS RIGHT SEATAC SHOWED
SNACKS SOUTHERN TIP TOLD VERY
WANNA YEAH

Although we have used much of the training data in developing the grammar, we have mainly focused on the annotated data. For the annotated data, we can compare our answers with the reference answers and determine whether we have parsed the sentence reasonably. This is consistent with the goal of our system, which is to extract all relevant information from the utterance. For the unannotated data, we need to manually look at either the output of the parser or the words missed to decide if the parse is reasonable.

We had on overall error rate of 5.8% on the November 1992 ARPA evaluation, and an error rate of around 4% on the ATIS2 annotated data. However, our error rate on the ATIS3 annotated data was around 18% for a nearly identical system. In the November 1992 ARPA evaluation, we had an error rate of 5.6% on the Class A set and an error rate of 6.1% on the Class D set. However, 75% of the Class A errors and 69% of the Class D errors in the evaluation set were caused by the lack of grammar coverage. We do not have exact results, but believe that the overwhelming number of errors in the training set are still caused by lack of grammatical coverage.

We try to generalize the grammar to parse strings that are syntactically similar to the utterances in the training data. In our current system, this is achieved by manually adding synonyms and using other completion techniques. We add other words that are related to the words in the training corpus, for example, if we see **Monday** in the training corpus, we add the word **Mondays** as well as other days like **Tuesday**. If we had been more thorough in our completion, we would have correctly parsed the sentences from the December 1993 evaluation that we mentioned above. The process can be vastly improved by automating this completion process based on synonyms, antonyms, plurals, possessives and other semantic classes. We could also use morphemes to derive new words from the words used in training data.

3.2. Recognition Language Model

The lexicon used by the recognizer is initially based on the training data. We augment this lexicon using completion techniques described above, as well as adding the words from the grammar used by the parser. We also added many city names that are not in the database. The recognition dictionary contained 2924 unique words plus 10 nonspeech events. Currently, we allow nonspeech events to occur between any words, like a silence. Since we have added words to the lexicon which were not observed in the training data, we need to generate a bigram with appropriate probabilities for these words. Initially, we used a backed-off class bigram. The class bigram used 1573 word classes and was smoothed with the standard back-off method. In generating the class-based models, the probability of a word given a class was determined by the unigram frequency of the words rather than treating them as equi-probable. We then compared this to a bigram language model created by interpolating two different bigram models: 1) a backed-off word bigram and 2) a backed-off class bigram. Our initial experiments on an unseen test set indicated that the perplexity decreased from 21.7 to 20.58 when we used an interpolated bigram instead of a class bigram. When the recognizer was run on this test set (516 utterances) with the two language models, word error rate was reduced from 11% to 10% by using the interpolated bigram (compared to the class based). This is and error reduction of about 9%.

We then tried to use the parser grammar to help smooth the bigram. The hope was to improve the match between the language models used during recognition and parsing and to get a better estimate of the probabilities of unseen bigrams. A word-pair grammar was generated from the parser grammar, and probabilities added by assuming equi-probable transitions from a word to all of its successors. This should give a better probability to a bigram which had not been seen, but was acceptable to the parser, than to an unseen bigram not allowed by the parser. We then interpolated the class-based, word-based and grammar-based bigrams. However, when evaluated on a test set, a recognizer using this model was not significantly different than using the interpolated word and class bigrams.

The trigram language model used the same word classes as the bigram, also smoothed by the back-off method, but not interpolated with a word-level model. We felt that we did not have enough data to train word trigrams.

4. PARSING

As described earlier, the system is implemented as a chart parser for slot fillers. As tokens (semantic phrases) are recognized, they are added to frames to which they apply. This process naturally produces partial interpretations. Words which don't fit into a interpretation are left out. In many cases the partial interpretation is sufficient for the system to take the desired action. If not, it still provides a good basis to begin a clarification dialog with the user. The system can give meaningful feedback to the user about what was understood and prompt for the most relevant missing information. The algorithm also produces multiple interpretations. Many different frames, and several different versions of a frame, are pursued simultaneously. In earlier papers [4], we have described some of the heuristics used by the parser to select the best interpretation for an utterance. If the score is below a certain threshold (based on the number of words in the utterance) it does not generate any parse.

The implementation of a chart mechanism had other advantages. The system does not allow overlapping phrases in a frame, two different slots could not use a given word in the input. The previous version of the system used a subsumption process to favor matching the longest strings possible, but did not keep substrings of the longest string. This led to a phrase overlap problem if one slot could start with the same word that could end another slot in the same frame. For example in the utterance "Show flights from Boston", if "show flights" matched one slot and "flights from Boston" matched another slot, both could not be assigned to the same frame since they overlap on the word "flights". The grammar writer could avoid the problem, but had to be careful to do so. The chart algorithm efficiently produces and keeps substrings. Now, it is not neccessary to bias for longer strings, the overlap problem is solved by producing both parses at very little extra cost. In the example, we would produce "show flights" "from Boston" and "show" "flights from Boston", assuming the grammars allowed these phrases.

5. Alternate Interpretations

Sometimes heuristics fail to identify the best interpretation, since they do not use additional information available in the context. We found it helpful to sometimes pursue alternate interpretations, that is interpretations which were not the best according to the heuristics used by the parser. In this section, we will describe when alternate interpretations are needed.

One of the design decisions in our system was to use task knowledge in the backend instead of in the parser. The parser only uses domain independent heuristics. This sometimes leads to ambiguities. For example, **Show me the fares** could refer to fares for flights or fares for ground transportations. In general, heuristics correctly identify the parse. However, consider the following two sentences:

> I NEED FARES FROM MILWAUKEE TO LONG BEACH AIRPORT **(q0g0b7ss)**
> I NEED FARES FROM MILWAUKEE TO GENERAL MITCHELL INTERNATIONAL AIRPORT

These sentences are syntactically identical. The user is most likely asking for flight fares in the first sentence. In the second sentence, the user is most likely asking for the cost of ground transportation. However, the system needs domain knowledge to make these distinctions; it needs to know that Long Beach airport is in California, while Generel Mitchell International is in Milwaukee. Since the parser has no domain knowledge, it cannot parse both of these sentences correctly. However, the backend has domain knowledge and notices that one of the parses is incorrect.

We address this problem by generating a beam of interpretations. The parser still produces the single best interpretation, but keeps track of a number of other interpretations. Whenever the backend notices a problem, it asks the parser for another interpretation. The parser then selects the next best interpretation. However, the score of the new interpretation must be within a certain threshold of the best score.

We next look at the parses for the above mentioned sentences:

> **i need fares from milwaukee to general mitchell international airport**
> [transport_select_field] [list_spec] I NEED [ground_fare] FARES [city_airport] FROM [city] [cityname] MILWAUKEE TO [airport_name] GENERAL MITCHELL INTERNATIONAL AIRPORT

> **i need fares from milwaukee to long beach airport**
> [transport_select_field] [list_spec] I NEED [ground_fare] FARES [city_airport] FROM [city] [cityname] MILWAUKEE TO [airport_for_city] [city] [cityname] LONG BEACH AIRPORT
> **ERROR correction: Frame Changed**
> [flight_field_list] [list_spec] I NEED [flight_fields_exist] [fare] FARES [flight_type] FROM [depart_loc] [city] [cityname] MILWAUKEE TO [arrive_loc] [city] [cityname] LONG BEACH

This error correction mechanism was used twice in the December 1993 evaluation set, and both times it worked correctly.

6. RESULTS AND CONCLUSION

In the December 1993 ARPA evaluation, systems from a number of ARPA sites were evaluated. The evaluation has three parts, speech is processed by the recognizer (SPREC), transcripts of the utterances are processed by the NL portion of the system, and then the speech input is processed by the entire system. Processing transcripts shows the NL coverage of the system and gives a baseline measure of how well it would do if recognition were perfect. Processing starting with the speech input then shows how much performance is lost due to recognition errors. The evaluation measures the error rate for each process. The measure used for the SPREC test is word error rate. This is the sum of all insertion, substitution and deletion errors. The NL and SLS sytems are scored on whether they produced a correct answer from the database. For these tests an answer is either correct (if it agrees with annotated database responses) or incorrect (if it does not). We had an error rate of 4.4% for SPREC, 9.3% for NL and 13.2% for SLS. These were the best results reported for the evaluation. So, for 9.3% of the transcript input, our system produced a wrong answer (there is no indication of whether it was close). The word error rate of 4.4% for the recognition gave a sentence error rate of 22%. That is, 22% of the utterances contained at least one recognition error. This number becomes 20% if class X utterances are excluded. There were 773 non-X utterances in this test set, so approximately 154 of the sentence hypotheses produced by the recognizer contained errors. Approximately 30 of these led to an error for the SLS system (when the transcript had been correctly processed).

7. ACKNOWLEDGEMENTS

This research was sponsored by the Department of the Navy, Naval Research Laboratory under Grant No. N00014-93-1-2005. 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 U.S. Government. We thank Raj Reddy and the rest of the speech group for their contributions to this work.

References

1. Deborah A. Dahl, Madeline Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the DARPA Human Language Technology Workshop*, March 1994.
2. Xuedong Huang, Fileno Alleva, Mei-Yuh Hwang, and Ronald Rosenfeld. An overview of the SPHINX-II speech recognition system. In *Proceedings of the DARPA Human Language Technology Workshop*, March 1993.
3. Sunil Issar and Wayne Ward. CMU's robust spoken language understanding system. In *Proceedings of Eurospeech*, September 1993.
4. Wayne Ward. The CMU air travel information service: Understanding spontaneous speech. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 127–129, June 1990.