

SPEECH RECOGNITION FOR A TRAVEL RESERVATION SYSTEM

Hakan Erdoğ an

IBM TJ Watson Research Center
PO Box 218 Yorktown Heights, NY 10598
erdogan@us.ibm.com

ABSTRACT

In this paper, we present our work on speech recognition for a spoken dialog system for automatic travel reservations. The system can receive speech input from an analog line or from IP telephony through a web site. The system uses speech recognition, natural language understanding, air travel database access, dialog management, natural language generation and speech synthesis technologies to perform the task of an automated travel agent. Language modeling for a mixed-initiative spoken dialog system is a challenging problem. We explore class-based n -gram language models (LMs) for this domain. Details for designing LM classes and assigning non-uniform probabilities within the classes are provided. We point out disadvantages of class-based LMs and introduce compound word vocabularies to overcome the problems observed. Dialog state dependent LMs are explored which enables incorporating semantic context into the language models. We also analyze the use of embedded context free grammar objects within LMs and point out advantages and disadvantages of using them. The resulting LMs are implemented and evaluated using IBM telephony speech recognition engine. Our efforts are shown to decrease the word error rate from 24% to 17% on an evaluation testset.

Keywords: speech recognition, spoken dialog systems, language modeling, context free grammars

1. INTRODUCTION

Automation of travel reservations over the web has been a successful business model for many internet companies. This is a step towards replacing human travel agents with intelligent machines. The next step towards a more human like experience in automated travel reservations is to use automatic speech recognition and natural language understanding to interact with a machine and book the travel reservations using a more natural communication medium, namely speech.

There exists commercial products that perform speech recognition and dialog management for just this purpose. However, current commercial systems have limitations. The callers have to know the system and they

cannot say anything they want and have the system understand their needs. Also, the commercial product does not provide a free dialog user interface, but the users are directed by the system as to what they can say at a particular state in the dialog. The responses are modeled by context free grammars (CFGs) and hence the responses have to be in a specific format.

DARPA initiated a program called “Communicator” which aims to provide a more natural dialog system for travel reservations as compared to current commercial products. A key point in the communicator program is that the systems have to have a user-interface which is mixed-initiative. Ideally, the systems should be flexible to accommodate the needs for both advanced and beginner users and they should let the user pick what they want to say at any point in the dialog. More information about this task can be found in [1, 2, 3, 4].

IBM communicator travel reservation system can be described as an automated travel agent that helps callers make airline reservations using natural language and dialog. The speech source can be POTS telephony or digital IP-based telephony through an e-commerce web site. The goal in the Communicator project is to design a system that behaves as close to a human operator as possible. The system combines speech recognition, natural language understanding, dialog management, database access, language generation and speech synthesis to perform the desired task. It is much simpler to design a speech recognition system that has limited input and uses fixed grammars as possible input speech. However, the challenge comes from the fact that, we would like more natural language to be used as input speech.

In this paper, we report our progress on the speech recognition portion of the IBM communicator system. Our work can be divided into two areas, acoustic modeling and language modeling. We report our detailed acoustic modeling experiments in [5], which also includes brief description of our language modeling work. We will focus on language modeling portion of the recognition system in this paper. Some of the improvements mentioned here were included into the live deployed sys-

tem, some are in developmental stage and are planned to be included in the system in the near future.

2. SYSTEM DESCRIPTION

The architecture of IBM communicator system has a Galaxy-II compliant [6] hub architecture. It is similar to the one described in [7]. The system is composed of a number of modules that interact with each other through the hub. The hub sequentially passes tokens to each module and maintains state of the system. There are modules such as recognizer, TTS, parser, dialog manager and language generation. The architecture is shown in Figure 1. It is possible to replace the telephony interface with IP interface to acquire audio directly from a web site. This might make it possible to use the system over the internet instead of the current telephony interface.

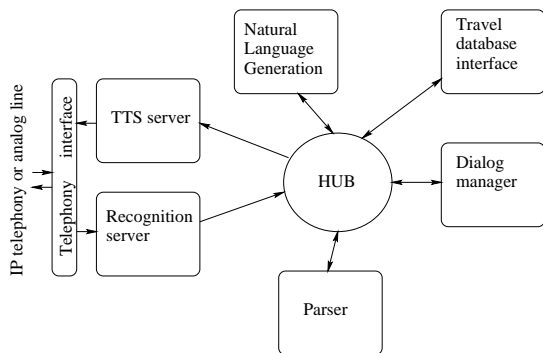


Figure 1: Travel reservation system architecture.

A typical interaction between the system and a user can be as follows:

System: Hello there, please tell me about the first leg of your trip. For example you can say, I want to fly from Los Angeles to Chicago leaving next Tuesday.

User: I want to fly from Beijing to Los Angeles.

System: Flying from Beijing China to Los Angeles International Airport on what date?

User: Tomorrow.

System: What time would you like to leave on Saturday May fifth?

User: Anytime please.

System: I see a couple of flights. (the system goes on to give details about each flight).

User: I would like the first one.

System: ...

User: ...

The system allows users to give more information in response to each prompt. For example, when asked for

date, the user could have said “Anytime tomorrow” to give both date and time in response to the date question. The user could also have said “No, I am going to Las Vegas not Los Angeles” to correct previous recognition and understanding errors if any. This is due to the mixed-initiative nature of the system which allows more natural interaction between the system and the users.

We focus on improving the speech recognition module in this work, specifically language modelling for speech recognition.

3. SPEECH RECOGNITION

Automatic speech recognition has been a research area for 30 years. Significant improvements have been made in computational speeds and speech recognition technology to make it an interesting area of business as well. Main goal in speech recognition is to have a machine recognize what word is being spoken by an arbitrary person. Applications range from desktop dictation, various telephony applications such as voice portals, directory dialers to embedded speech recognition applications such as command and control in a car. Initially, studies were done on limited vocabulary isolated speech recognition. However, recently large vocabulary continuous speech recognition systems have been developed and are being made commercially available by several companies.

Fundamental equation of speech recognition is as follows:

$$w^* = \operatorname{argmax}_{\text{words}} P(\text{word}|\text{speech signal}),$$

that is to find the most likely word given the acoustic data. We can rewrite this equation in the following form which is easier to tackle:

$$w^* = \operatorname{argmax}_w P(w|x(t)) = \operatorname{argmax}_w P(x(t)|w)P(w),$$

where $x(t)$ represents the acoustic signal. The first part of the objective function $P(x(t)|w)$, the probability of generating acoustics $x(t)$ by speaking word w , is handled by the acoustic model. The second part $P(w)$ is the probability of the word w itself in the language. This probability depends on the context of the speech. Language modeling enables us to estimate this probability $P(w)$. The score for each word is determined by adding the log probabilities of acoustic and language models with appropriate weights. Evaluating the score for each word in a language is impractical in large vocabulary recognition. There are time alignment issues as well for continuous speech recognition. These issues are handled by efficient search strategies such as Viterbi search and A^* search. In IBM, we use a stack decoder

(speech recognizer) which is based on A^* search. Next, we explain acoustic modeling briefly and detail our work on domain specific language modeling for travel reservations.

4. ACOUSTIC MODELING

State of the art acoustic modeling is performed via Hidden Markov Models (HMM) in speech recognition. A vector of features are extracted from acoustic signals every 10 ms. Typically MFCC features are used. Each phoneme features are modeled by a three state HMM where states correspond to begin middle and end of a phoneme. It is also observed that each phoneme behaves differently in different contexts. Phonemes in different contexts are modeled separately. Each state is modeled by a probability distribution parameterized as a mixture of Gaussians. The parameters of an HMM model are these Gaussian means, (co)variances, mixture weights and transition probabilities between contiguous states. There is a well known training algorithm for HMMs called forward-backward algorithm (Baum-Welch) in which acoustic data as well as the script corresponding to the acoustic data are used to train the model parameters.

We built domain specific acoustic models for our travel reservation system. The training data were collected from internal calls made to the IBM communicator system, as well as generic telephony data from different domains. Since the travel domain speech recognition is required to recognize city names and the training data did not contain all possible city names, it was necessary to add generic telephony data to increase coverage of the acoustic model for all different cities and airports. We explain our work on language modeling next.

5. LANGUAGE MODELING

Calculating the probability of a word in a given context is a problem that involves many parameters. The probability of a word in a certain position in a sentence can depend on the general topic being conveyed, the words that occur before and after the word, the syntax of the sentence, the meaning (semantics) of the sentence, *etc.*

Some speech recognition systems use context free grammars (CFGs) to constrain what the user can say to the system. CFGs define a formal language representable by a finite state machine (FSM) which only allows certain sentences to be spoken. The word sequences that do not correspond to the grammar receive a probability of zero. The probability of an acceptable word sequence is determined by probabilities assigned

to each path in the grammar. The grammars also simplify the search problem in speech recognition by eliminating many possibilities during search. However, the application of CFGs in spoken language applications is limited. Another method to compute word sequence probabilities are called n -grams.

To simplify the word probability estimation problem, in n -gram language models, the probability of a word w_i in a context is assumed to be dependent on the previous n words only:

$$P(w_i|\text{context}) \approx P(w_i|w_{i-n+1} \dots w_{i-2}w_{i-1}),$$

where w_j represents the word in position j . The trigram probability $P(w_i|w_{i-2}w_{i-1})$, and bigram probability $P(w_i|w_{i-1})$ are mostly used. The unigram probability is $P(w_i)$ assuming no context information. The maximum likelihood estimates for n -grams are found by counting the occurrence of a specific n -gram in training data and dividing by the total number of n -grams. Usually, trigram language models are used where the probabilities are smoothed with bigram and unigram probabilities for robustness and to avoid zero probabilities.

$$P(w_i|\text{con.}) \approx \lambda_3 P(w_i|w_{i-2}w_{i-1}) + \lambda_2 P(w_i|w_{i-1}) + \lambda_1 P(w_i),$$

where λ 's are estimated from held-out data and are dependent on counts of the word sequences w_{i-1} and $w_{i-2}w_{i-1}$ in the training data.

The reason for using a small value of n is that the number of parameters increase exponentially with n and that the training data becomes sparse to estimate these parameters reliably. Thus, standard smoothed trigram LMs do not model longer correlations between words, neither do they take advantage of linguistic structure or knowledge. However, in most cases they provide satisfactory performance given enough training data and appropriate smoothing.

5.1. Class-based Language Models

Trigram language model is very powerful if there is enough training data that covers a big portion of possible sentences in a domain. However, most of the time, the training data does not cover all possible sentences and words. In this case, word trigram language models do not accurately represent the language. Furthermore, some words are logically connected and it makes sense to tie their probabilities together instead of relying on the individual training data counts for these words. Thus, we "regularize" the maximum likelihood estimation of trigram probabilities by using our prior knowledge. This approach increases coverage and context richness. In travel domain, there are plenty of

classes of words, such as months of the year, days of the week, *etc.* We use class-based language models to represent the tying among these similar words. Each word w is assigned to a class $C(w)$, where the class can be a singleton set¹ or a multiple element set. In this case, the word sequence probability for a class-based trigram becomes:

$$P(w_i) = P(w_i|C(w_i)P(C(w_i)|C(w_{i-2})C(w_{i-1}))),$$

where $C(w)$ represent the word class for word w and $P(w_i|C(w_i))$ is the within class probability (or weight) of word w_i in the class $C(w_i)$. For certain classes, we use uniform within class probabilities and for some others we use non-uniform weights.

In our language model for the travel reservation system, we have 42 classes representing cities, states, airports, airlines, months, years, days in a month, hours, minutes, days of the week as well as some other classes.

We have nonuniform within class probabilities for cities and they are obtained as a weighted combination of unigram counts in training and airport volume data. So, big cities with higher airport volume receive a higher probability as compared to a small city. For states, we have obtained nonuniform weights by adding up airport volume data for airports within each state. For airlines, we also have used a similar technique where each airline probability is proportional to the airline passenger volume data. For other classes, we either used uniform or heuristic weights. We have iterated this process to achieve the best results. For example, we had a class of years which had uniform weights for each year from 1990 to 2010. After observing some errors, we changed the weighting to be nonuniform and made current and following years (1999, 2000, 2001, 2002) more likely since it is highly unlikely that one would call the system and want to reserve a flight for 3 years later. We achieved a significant error reduction with this kind of small modifications.

LM classes are good for increasing coverage and enriching the context. However, we have observed a problem with the classes we used which was related to cities and states. In travel reservations, when people talk about their travel plans, they often pronounce the city and state name one after the other such as “DALLAS TEXAS”. So, if trigram training is used the bigram “DALLAS TEXAS” has a high probability whereas “DULUTH TEXAS” has a low probability. However, if class trigram is used with [city] and [state] classes, “[city] [state]” has high probability regardless of the individual city or state. This problem causes speech recognition errors such as “SIOUX CITY OHIO” (instead of “SIOUX CITY IOWA”) and “DALLAS MINNESOTA”

(instead of “DULUTH MINNESOTA”) which would have been highly unlikely if a regular trigram had been used. These speech recognition errors are highly undesirable for the travel reservation system since it is hard for the dialog system to recover from these type of errors. The same kind of problem occurs for “[city] [airport]” pairs. To overcome these significant problems, we used the compound word approach which we explain next.

5.2. Compound Words

Trigram LMs work on words directly. It is advantageous to glue frequently occurring word sequences together and assume that they always appear together for the purposes of the LM. We concatenate such words, *e.g.* the word sequence “SAN FRANCISCO” can be glued together to form “SAN_FRANSISCO”, since it is highly likely that the word “SAN” will appear together with “FRANSISCO”. In our system, by default the city names are combined together such as “NEW_YORK” and “LOS_ANGELES”. The advantage comes from the facts that (1) we can use a larger word history (in trigrams) when we consider this sequence as one word, (2) longer words have a higher chance of being recognized correctly since number of confusable words decrease as the words get longer and acoustically matching a longer word is harder than matching short words. So, if the decoder gets a high acoustic likelihood for a long word, it is most probably the correct word.

There are some disadvantages for using compound words. (1) Need to generate pronunciations for the compound words for which there might be many variations depending on how many pronunciation variants there are for each individual word. (2) Including possible silence in between words also increases lexicon size. (3) When the words are glued, if one of the constituent words occur in another context, they become very unlikely (since their unigram probability falls) and recognizing them in other contexts becomes more difficult.

It might also be advantageous to tie other frequently occurring word sequences together. There is an automated method for determining compound words from training data [8]. This method uses the mutual information between words to determine whether to combine them or not. For the travel reservation domain, automatically generated compound words did not improve recognition rate [4].

We have used compound words to overcome the problem we faced with the language model classes [city] and [state]. We glued together the word sequences corresponding to “[city] [state]” allowing only the correct combination such as “DALLAS_TEXAS” or “LOS_ANGELES_CALIFORNIA”. These compound words

¹In which case we can think that the word is itself a class

were then put in a LM class of their own [city_state] to increase coverage. This method helped reduced the word error rate (WER) significantly. The error corrections were at important places for the language understanding and major errors in city names can be avoided using this method. Similarly, we glued together “[city] [airport]” pairs such as “NEW_YORK_LA_GUARDIA” to aid in recognition as well. These were put in an LM class of their own as well.

We also have other compound words that were formed by choosing a subset of the word sequences generated by the method in [8]. These include compounds like “I_WOULD_LIKE_TO_FLY” and “THAT’S_O.K.” and many others.

Because of the third disadvantage of compound words, it is beneficial to interpolate trigram probabilities with the ones obtained without the compound words present. The best result is obtained when regular trigram LM and compound-word trigram LM are interpolated with equal weighting between the two.

5.3. Dialog State Dependent Language Modeling

Dialog manager in the travel reservation system can provide feedback to the speech recognition module about the state of the dialog. We use a form based dialog manager (FDM) which fills out forms according to user input. At a given point in the dialog, the FDM maintains the state of the dialog and prompts for new information from the user to finish the transaction. This information is useful, because it can be used to help in modeling the language of the response. For example, if the dialog manager is asking about target destination, it is highly expected that a city name will be in the response. Similarly, if a question about the departure time is being presented, the response most likely will contain time information. The dialog state can also be inferred from the system prompt indirectly by using some rules. State dependent trigram probability is given by:

$$P_S^{\text{tri}}(w_i) = P(w_i | w_{i-2} w_{i-1}, S)$$

where S represents the current dialog state. We have dialog states in our system corresponding to departure and arrival dates and times, departure and arrival city, airline and confirmations. To use this information, we built language models trained from a corpora of user responses recorded at a specific state. We interpolate these language models with the base trigram LM to obtain a different LM for each state. This is done since we do not have enough data for each state to train the LM and the answers do not have to comply with the questions all the time since this is an open dialog system.

Dialog state feedback for language modeling improves recognition rate about 5% relative.

It should be noted that, since our training data for each state was limited, it is better to use a bigram language model for each state in order not to overtrain the system. However, if enough data can be obtained for each state, trigram LM can be used.

5.4. Embedded CFGs

Context free grammars (CFGs) represent a finite language which allows only certain word sequences. They could be represented by a finite state machine. CFGs are widely used in language processing for parsing sentences. CFGs with probabilistic paths through the grammar are called stochastic CFGs (SCFGs). DARPA communicator is a free-form mixed-initiative dialog system, so direct application of context free grammars is not appropriate. However, within the utterances, there are well structured portions (phrases) such as the parts that are describing the arrival or departure dates, times and places. IBM speech recognition engine has the recently added capability of using embedded SCFG objects within the language models [9]. The portions of speech that matches a grammar are tokenized with a special token and a trigram LM is trained with the new tokens in place. During decoding, if a word is hypothesized that occurs in the first position for the grammar, then the grammar is hypothesized and that path is scored using the grammar. This scoring is incorporated into IBM’s stack decoder. This process is equivalent to a language model probability as follows:

$$P(w_i) = P(\dots w_i | G_j) P(G_j | G_{j-2} G_{j-1}),$$

where G_j represents the grammar object that includes a path that contains the word w_i and the trigram history is considered with the grammar objects². The function of embedded CFGs is similar to using compound words and LM classes with non-uniform within class probabilities. So, it has the advantages listed above for the compound word approach. The difference is that with the CFGs, there is no need to define compound words and thus it is possible to capture a wider variety of word sequences by minimal effort. Another advantage is that a longer silence can be inserted between individual words and they will still be recognized. However, when highly inclusive grammars are used, there are some disadvantages as we explore in the following.

Since we already worked on the compound words and classes as mentioned before, we designed new embedded CFGs to have broader coverage than the LM classes, such as a place grammar which contained expressions such as “FROM [city] TO [city]” as well as

²We allow single word grammars as well.

“FROM [city]” and “TO [city]” for all cities. We also designed different CFGs for dates and times which allowed expressions such as “MONDAY SEPTEMBER THIRD TWO THOUSAND ONE” and “TEN A.M. IN THE MORNING” and many more. The probabilities for paths in the grammars were assigned using common sense. Using these CFGs, a possible parsing of a travel domain sentence is shown

I WANT TO FLY FROM NEW_YORK
place

NEXT TUESDAY IN THE MORNING.
date time

Although this approach seemed promising, we were disappointed to see that the error rate increased when the embedded CFGs were used. We have determined the following reasons for the failure of this approach:

- The weights (probabilities) within the CFGs are assigned subjectively and this is not usually appropriate for speech recognition purposes. Trigram probabilities capture the word probabilities better. Ideally, we would like to train these weights using a training corpus, which we are currently exploring.
- The grammars should be designed to represent same logical entities where the context is expected to be exactly same for each path in the grammar. For example, if “ON DELTA” is in the same grammar with “A DELTA FLIGHT” that hurts the performance since the sequence “FLY ON DELTA” is more likely than “FLY A DELTA FLIGHT” but the CFG inherently assumes they are the same. So, the grammars should be designed keeping this point in mind.
- If a word appears in two or more grammars, or a word appears in different paths in a grammar, the LM probability of the word decreases as compared to the n -gram LM. The reason for this is that when computing the LM probability of the current word, only one path through the grammar is considered. So, grammars should be designed to make sure that each word appears in only one grammar and that it never (or rarely) occurs outside that grammar.

Because of the above problems, best results are obtained when embedded CFG LM probabilities are interpolated with regular smoothed trigram LM probabilities. We used equal weighting between the two.

We revised our CFGs after observing the above points, and redesigned new embedded CFGs corresponding to dates, times, cities and “city state” pairs, airports and

airlines. We experimented with different grammar designs. However, the best result we obtained was only slightly better than the class-based LM in some testsets and worse in others. We are exploring new grammar designs and working on ways to estimate within grammar weights more accurately from training data. We present our experimental results next.

6. EXPERIMENTAL RESULTS

The acoustic and language models were developed and tested in the IBM communicator system which is used for flight reservations. We trained the acoustic models using 380 hours of travel domain speech from about 4000 speakers and 250 hours of general telephony domain speech. Language models were trained using 100K sentences from travel domain collected from various sources. The numbers in the corpus were tagged using the method described in [10] to help in language modeling. There was a held-out set of 18K sentences used to optimize trigram smoothing parameters.

Our main test set is composed of a subset of the calls that the IBM Communicator system received during the NIST evaluation of various DARPA communicator systems in June 2000 (1173 utterances out of 1262 received).

The baseline model is the IBM speech recognition system used during the June 2000 NIST evaluation. Acoustic model (old AM) is a generic telephony system that is trained from a wide variety of telephony data (600 hours of speech). Language model (old LM) is trained using 90K sentences from travel domain and has LM classes and no compound words except city names.

As mentioned above, we added 10K more sentences to the LM training data, redesigned the classes and added new classes and compound words as described in Section 5. A few such LM’s were designed. The best one (“new LM”) is chosen among all.

Table 1 shows the overall improvement in word error rate due to the new acoustic and language models. The results are on the main testset.

Acoustic Model	Language Model	WER
old AM	old LM	23.70%
old AM	new LM	20.63%
new AM	old LM	19.03%
new AM	new LM	17.79%

Table 1: Comparison between old and new, AM and LM. Overall a big reduction is shown.

As shown in Table 1, we achieved a significant reduction in error rate with new acoustic and language

models. Addition of compound words and adjustment of non-uniform within class weights improved error rate about 15% relatively.

To test the dialog state dependent language modeling, we used six dialog states and all dialog feedback states were mapped to these six different states. For each state, we used about 500-1000 state dependent sentences for training. We did not have enough data for dialog state dependent LM training. We believe dialog state feedback can improve performance more if more training data is collected.

The effect of the dialog state feedback and embedded CFGs are shown in table 2. Here, we show the effects on 3 testsets. "main" is the same as the one in table 1. The other two are decodings of the data received by 2 other sites during the evaluation. For anonymity, the names of the sites are not mentioned.

It is shown that the dialog state feedback reduces the error rate by about 5%. For other sites, the dialog state is estimated from their system prompts (for testing purposes) using some rules since the real feedback information was not available. For embedded CFG LM result in column three, no dialog state feedback is used. Embedded CFGs increased error rate for IBM received calls, but reduced error rate for the other two sites. This could be due to the different nature of calls that were received by different sites. Some sites have a more directed dialog system that encourages more structured responses which might explain why embedded grammars helped more for the other sites.

Language Model	main	test2	test3
new LM	17.79%	19.70%	27.74%
Dialog state dependent	16.91%	19.15%	27.45%
embedded CFGs	19.19%	17.23%	27.23%

Table 2: Comparison showing LM with NLU state feedback and embedded grammars. Same acoustic model is used throughout.

7. CONCLUSION

Our innovative work in language modeling reduced speech recognition errors a substantial amount on a travel reservation spoken dialog system. Most reduction is obtained by carefully retraining language models using domain specific data and cleverly increasing the LM coverage using classes. Including "[city] [state]" compound words in the vocabulary were shown to be effective for better performance. Dialog context dependent language modeling improved error rate consistently over all testsets. However, using embedded CFG objects in

n -gram language models did not uniformly improve the error rate. Our work is in progress in several directions (1) Better design of embedded grammars, (2) Rescoring lattices using more elaborate LM techniques using CFGs, and (3) automatic training of within grammar weights using a training corpus.

8. ACKNOWLEDGEMENTS

The author would like to thank Yuqing Gao, Andy Sakrajda, Adwait Ratnaparkhi, Kevin Smith, Xiaoqiang Luo, Kishore Papineni, Todd Ward, Salim Roukos, and many other members of IBM HLT department for designing, building and developing other parts of the dialog system.

REFERENCES

- [1] E. Levin et al., "The AT&T-Darpa communicator mixed-initiative spoken dialogue system," in *International Conference on Spoken Language Processing*, 2000.
- [2] B. Pellom, W. Ward, and S. Pradhan, "The CU communicator: An architecture for dialogue systems," in *International Conference on Spoken Language Processing*, 2000.
- [3] A. I. Rudnicky et al., "Task and domain specific modelling in the Carnegie Mellon communicator system," in *International Conference on Spoken Language Processing*, 2000.
- [4] A. Aaron et al., "Speech recognition for Darpa Communicator," in *Proc. IEEE Conf. Acoust. Speech Sig. Proc.*, 2001.
- [5] Y. Gao, H. Erdogan, Y. Li, V. Goel, and M. Picheny, "Recent advances in speech recognition system for IBM Darpa communicator," in *European Conference on Speech Communication and Technology*, 2001.
- [6] S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue, "Galaxy-II: A reference architecture of conversational system development," in *International Conference on Spoken Language Processing*, 1998.
- [7] K. Davies et al., "The IBM conversational telephony system for financial applications," in *European Conference on Speech Communication and Technology*, 1999.
- [8] G. Saon and M. Padmanabhan, "Data-driven approach to designing compound words for continuous speech recognition," in *Automatic Speech Recognition and Understanding Workshop*, 1999.
- [9] M. Monkowski, "Embedded grammar objects within the language models," Technical report, IBM, 2001.
- [10] X. Luo and M. Franz, "Semantic tokenization of verbalized numbers in language modeling," in *International Conference on Spoken Language Processing*, volume 1, pp. 158-61, 2000.