Conversation Machines for Transaction Processing

Wlodek Zadrozny, Catherine Wolf, Nanda Kambhatla, Yiming Ye

IBM T.J Watson Research Center,
30 Saw Mill River Road,
Hawthorne, NY 10532
{wlodz, cwolf, nanda, yiming}@watson.ibm.com

Abstract

We have built a set of integrated AI systems (called conversation machines) to enable transaction processing over the telephone for limited domains like stock trading and banking. The conversation machines integrate the state-of-the-art technologies from computer telephony, continuous speech recognition, natural language processing and human-computer interaction. Users can interact with these systems using natural language to process simple transactions. We are currently installing a prototype conversation machine at a customer site (a large bank), while continuing research on each of the modules mentioned above and their integration.

In this paper, we describe the architecture of conversation machines and explain the design choices related to natural language dialog design, speech recognition errors, and human-computer interaction. We also discuss our experience with the new "market-driven research" methodology currently being tested at our company, of which the conversation machines project is an example. Our experience suggests that with this new methodology we can build integrated natural language dialog systems, even when working with error-prone recognition engines and imperfect grammars, by designing the dialog flow to reduce the likelihood of errors, and to enable quick error recovery. In this process, having a customer allows us to make more realistic design choices.

1. Introduction

In the past few years we have developed a set of integrated artificial intelligence (AI) systems, called conversation machines, that allow users to communicate with a computer using spoken language; for example, to make inquiries, pay bills or transfer money from a bank over the telephone. Other tasks we have dealt with include stock trading, fast food ordering, managing appointments in an on line calendar, and payroll processing. All of these tasks involve an integration of computer telephony interfaces, continuous speech recognition, spoken language understanding in small, but non-trivial, domains, and huma n computer interaction.

In a new model of research at our company, we are installing a prototype Conversation Machine for banking at one of the large American banks, while continuing research on each of the afore-mentioned modules and their integration. This system differs from other such systems in the market with respect to

- the quality and range of the human factors engineering in the system and
- the diversity of the number of sentences accepted by the system (for instance, the natural language grammar of our system accepts over 50 million natural language queries, as opposed to a few thousands for most such systems in the market).

We are planning to conduct trials with customers of the bank in 2-3 months to continue the research on human computer interaction issues and natural language dialog issues.

The Conversation Machine for banking allows customers to interact with their bank accounts by telephone, using spoken language dialog. The customers can speak requests in regular English, and the system performs the requested actions, asking for additional information if necessary. The customers can use any English phrase to request information or perform an action without having to remember a menu, as illustrated in the following three dialog fragments:

I'd like to transfer some money to my checking account.
To make sure I understand the amount of money to transfer, please enter the amount…
3 5 0 0 0 #
I am about to transfer $350.00 from your savings to your checking account. Do you want me to do it?
Yes.
Transfer completed. Your new savings balance is $1011.00. Your new checking balance is $1100.67.

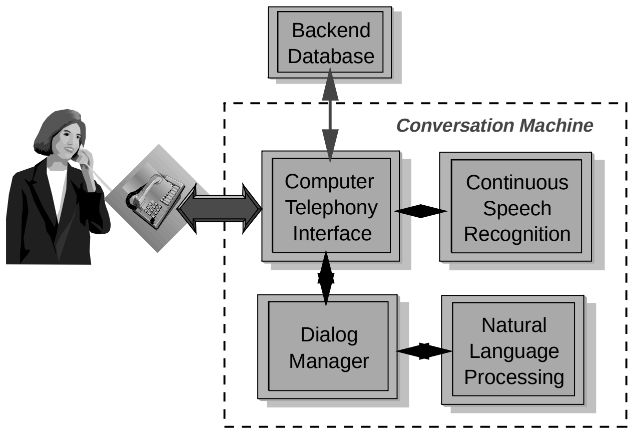
Pay my phone, insurance and electricity bills.
I am about to pay $224.00 for your phone, insurance, and electricity bills. Do you want me to do it?
Yes.
I have paid your phone, insurance, and electricity bills. Your new checking balance is $507.00.

Mastercard and insurance bills please.
Your MasterCard balance is $125.00. Your insurance balance is $43.00. What else can I do for you?
Pay them.
…

In the next section, we describe the architecture and components of a Conversation Machine. In section 3, we discuss some of the tools we built to support research and development on the Conversation Machine. We then describe some of the lessons we learnt during the research, design and field trials of the conversation machine and present our conclusions.

2. Architecture of Conversation Machines

In this section, we describe the architecture of a Conversation Machine and briefly describe its component units.


Backend Database
Conversation Machine
Computer Telephony Interface
Continuous Speech Recognition
Dialog Manager
Natural Language Processing

2.1 Architecture of a Conversation Machine

Figure 1 shows a schematic block diagram of a Conversation Machine (CM). A user interacts with the CM using a telephone. A computer telephony interface (CTI) unit answers the call and passes on the user's speech utterances to the continuous speech recognition engine. The speech recognition engine parses the speech utterances into sentences and passes them back to the CTI. The CTI then sends the recognized sentences to the dialog manager (DM) which, in turn sends the sentence to be analyzed by a natural language processing (NLP) unit. The NLP unit parses each sentence using either a *construction grammar* or a *semantic grammar* and returns an attribute value list called a *semantic signature* to the DM. (The details are given below). The dialog manager then determines the appropriate response to the user's speech utterance based on filled values of the semantic signature returned by the NLP unit. This action/response is communicated to the CTI unit which plays the corresponding prompts to the user along with (if necessary) some information obtained from a backend database (for instance, in response to a query asking for checking balance). The central role of CTI in the current system is due to the fact that it comes with all necessary communication devices.

2.2 System components

The Conversation Machine consists of four major modules: a Computer Telephony Interface, a Continuous Speech Recognition engine, a Natural Language Processing unit, and a Dialog Manager. The modules communicate through TCP/IP. The system is modular and runs on an RS/6000 model 390 under AIX.

Computer Telephony Interface: We used the IBM DT/6000, version 1.6 (to be replaced soon by version 2.1) system for providing a computer telephony interface (CTI). The CTI unit is responsible for

- answering a call by a user,
- sending the user's utterance to the continuous speech recognition engine,
- receiving a recognized sentence from the speech engine, and sending it to the dialog manager, and
- playing appropriate prompts to the user as directed by the dialog manager, by accessing the back-end database (if necessary).

Continuous Speech Recognition: Our system uses IBM Continuous Speech Recognition System for Telephony, VT-Tel, which supports large vocabulary speaker independent recognition. However, it works best if at any point of a conversation the active vocabulary does not exceed a few hundred words; therefore we have taken great care to design a dialog manager in a way that minimizes the active vocabulary at any point of conversation (by designing separate grammars with constrained vocabularies for nested subdialogs).

Natural Language Processing: We have built two versions of the natural language grammars. In the first version, the natural language grammars are represented as a collection of constructions (Goldberg, 1995), i.e. data structures where syntactic, semantic and pragmatic information are combined in one record. This representation has important computational advantages of being general, compact and object oriented. We also used semantic grammars designed to cover specific domains like telephony banking. These grammars are specific, compact, and easy to update and maintain. They can't be reused for new applications, but can be changed and updated by people with very limited computational linguistic skills. For space considerations, most of our discussion in this paper will focus on the version of conversation machines that use semantic grammars. For further information on construction grammars, see (Fillmore et al., 1988), (Goldberg, 1995), and (Jurafsky, 1992).

Dialog Manager: In our Conversation Machines, the Dialog Manager (DM) controls the flow of the conversation with a user. The DM is responsible for analyzing each sentence, deciding the next prompt(s) to be played, and the action to be taken, if any. Thus, the DM works in the conversation-for-action paradigm, i.e. it tries to find out which action to perform and what the parameters of the action are for each user sentence (after analysis by the NLP unit, in the form of a semantic signature). In other words, the DM is a finite collection of states and transitions plus a small number (<20) of global, contextual variables that help determine the next state and the utterance interpretation. If a whole utterance cannot be interpreted within a given context, the largest sub-part that makes sense is tried. Because of the possible ambiguities, there is a small set of rules (5 clauses) expressing the preferences in interpretation.

3. Building tools for the Conversation Machine

In this section, we briefly describe the tools we have developed for rapid development of conversation machines.

3.1 Speech grammar reviser

We have developed a speech grammar reviser- an interactive tool for constraining the sentences accepted by the speech recognition engine (Kambhatla and Zadrozny, 1998). Constraining the list of valid sentences reduces the likelihood of a sentence being mis-recognized. This is especially important for telephone based applications, where the recognition accuracy is not very high. Thus, the speech grammar needs to be both general enough to accept a large variety of sentences about the given application domain, and constrained enough to disallow all other sentences. Developing such tight grammars manually is a tedious and time consuming task. We have automated a large portion of this task through the grammar reviser tool.

Given an initial version of a speech grammar in Backus-Naur Form (BNF), the grammar reviser, in an interactive session with a grammar developer, identifies counter-examples, i.e. examples of sentences currently in the grammar but not appropriate for the given application domain. The grammar is then modified in such a fashion that the counter-examples are no longer accepted by the grammar. For each counter-example **c**, a parse tree is generated. For each non-terminal symbol **x** in the parse tree except for the TOP (or start) non-terminal, two new non-terminals are introduced which generate

- all the substrings generated by **x** except for substrings of **c** and
- the substring of **c** that **x** generates

respectively. The production rule that generated the sentence **c** from the TOP non-terminal is modified such that it no longer generates **c**. The above algorithm for grammar revision is described in detail in (Kambhatla and Zadrozny, 1998). Thus, a constrained version of the initial grammar is generated for use by the speech recognition engine.

3.2 Grammar Workbench

We used a grammar workbench developed at IBM TJ Watson Research Center to develop and maintain semantic grammars for our applications. The grammar workbench is a GUI tool designed to create and maintain grammars along with the corresponding dictionaries.

The dictionary is a list of all the words allowed by our dialog system along with a set of semantic categories that each word belongs to. For example, the following is an item in the dictionary for our stock trading system:
(cash *n_money ((tr (cash))) *n_accttype ((tr (cash)))).
Thus, the word "cash" belongs to the category *n-money (as in "I want to withdraw some cash") and also to the category *n-accttype (as in "charge it to my cash account"). The categories in the dictionary are carefully designed to reveal the hierarchical ontology structure of the application domain.

The grammar contains a set of production rules, each consisting of a syntactic rewrite rule followed by its corresponding compositional semantic translation rule. For example,
((teens -> (*numtens *numones)) (((tr 0) (+ (tr 1) (tr 2)))))
is a rule in our stock trading system that combines elements of two-word number names of category "*numtens" and "*numones" such as "fifty six" to create 2-digit number expressions of category "teens" such as "56" whose semantic translation "(tr 0)" is calculated by a Lisp function "(+ (tr 1) (tr 2))", where "(tr 1)" and "(tr 2)" refer to the semantic translations of "*numtens" and "*numones" respectively.

3.3 Chart Parser

We use a chart parser to parse recognized sentences into their semantic translations. For example, a sentence "what is my checking balance" generates a semantic translation "PROCESS = {[ACCTTYPE = {checking}, GETINFO = {balance}]}, SEM_TYPE = {info}, SYN_TYPE={np}]".
Since chart parsing is a standard NLP tool (see e.g. (Gazdar and Mellish, 1989)), we will not discuss any of its internal characteristics in this paper.

3.4 Linguistic to domain translation using semantic signatures

We perform the mapping from the semantic translations returned by the chart parser to a set of (domain specific) actions for the dialog manager using *semantic signatures*.
The semantic translation of a sentence can be considered to be a template, some parts of which are useful for identifying the application domain topic, and other parts are not. The parts that can be used to identify a domain topic form a signature for this topic. For example, a semantic signature for the domain topic "query about account balance" is "PROCESS = {[ACCTTYPE = {*}, GETINFO = {balance}]}, SEM_TYPE = {info}, SYN_TYPE={np}]". Thus, using semantic signatures, the

process of mapping sentences into domain meanings can be divided into two stages. First, a semantic grammar maps a large number of sentences about a given topic onto a few semantic translations. Then, a specification of a few (perhaps a handful) appropriate semantic signatures maps these semantic translations onto domain topics. The semantic signatures for an application are in a separate file. Thus, whenever the grammar is changed to incorporate new sentences to cover an existing action, only the signature file needs to be changed. Also, the code for the dialog manager module is reduced in complexity.

3.5 Dialog Manager Workbench

We built a workbench with a graphical user interface, called the Dialog Management Workbench (DMW), for building dialog managers. The DMW provides a high level fourth generation language, specifically designed for writing dialog managers. Code written in this language is translated into Java by DMW. Thus, the DMW enables a dialog manager developer to focus mainly on the flow of the dialog without worrying about specific programming language constructs that have no relation with the dialog. For example, in our system, a developer need not know the insides of a module that is used to extract the semantic signatures and to map the signatures into the domain topic. Another objective of the DMW is to force the dialog developer to write the dialog manager in a modular fashion such that the code is both readable and easy to modify. The DMW facilitates three levels of module nesting. For example, one of the modules in our stock trading system ("Conversation Loop" module) consists of the following states:

"GetCustomerInputSentence",
"IdentifyDomainTopic",
"CheckAndHandleTopicParameters",
"ExecuteTopic".

Each state of the above corresponds to a lower level module that contains the flow of the dialogs for this topic. For example, the "IdentifyDomainTopic" state contains DMW codes that accepts an input sentence, sends the sentence to the semantic parser, and obtains a domain topic. Thus, the DMW enables a developer to rapidly write a portable, modular and easy to read dialog manager.

4. What we learned

4.1 Guiding the user

As we developed the Conversation Machine for banking, we were aware that the state of the art for both telephone speech recognition, and natural language understanding was inadequate to allow users to talk to the system in a completely unconstrained way. The dialogs and prompts we developed are intended to guide and gently constrain the user to produce utterances which the system can understand.

Guidance for the next step – The system provides guidance for the next step at every stage. At the top level, the prompt *"What else can I do for you?"* lets the user know that the system is ready for the next request. When the system has only partial information for a dialog, it prompts the user for the missing information. For example, if the system hears "Pay my bills" it will ask the user "*Which bills would you like me to pay?"*. This allows the user to proceed in small steps if desired, thus reducing the disfluencies and consequent speech recognition errors which often characterize long utterances (Oviatt, 1995). This feature is also useful in the case where a recognition failure has caused the Conversation Machine to miss some information included in the original utterance. For example, if a user says "*Pay my phone bill*," and the speech recognizer fails to recognize "*phone*" the system will follow-up with "*Which bills would you like me to pay?"*. Thus, the user is guided to provide the missing information without repeating the entire request.

Our system also provides guidance which is helpful if a user is in a state inadvertently due to a mis-recognition. For example, if the prompt described above is unsuccessful in eliciting a valid response, the next prompt is "*Which bills? Please say electricity, phone, insurance, or MasterCard. To cancel, say cancel*." Reminding users how to get out of an unintended dialog can reduce frustration and help users get back on the right track.

Feedback for state –An important principle for the design of prompts in the Conversation Machine for banking was to always provide feedback for state in each prompt. For example, the prompt to enter the amount of money for a transfer is "*To make sure I understand the amount of money **to transfer**, please enter the amount…*". The inclusion of the phrase "***to transfer"*** lets the user know that he/she has entered the transfer sub-dialog. We were reminded of the importance of providing feedback for state when testing an earlier version of the system, which did not include this phrase for the transfer dialog. When users, who had asked to pay a bill, were in a transfer sub-dialog in error, they assumed they were being asked to enter an amount for bill payment. Only later in the dialog, did the mis-recognition error become apparent. Providing feedback for state in each prompt helps users detect errors and more quickly recover from them.

Progressive prompting –Another technique we use to help users recover from errors and avoid future errors is to give progressive prompting based on strategies that are likely to be successful and that users understand. As reported in (Wolf et al, 1997), some strategies that people spontaneously employed based on human-human communication are more likely to be successful with the Conversation Machine than others. Accordingly, our progressive prompting provides increasingly more direction with each consecutive error by suggesting generally successful strategies (see also Yankelovich et al., 1995).

For example, after the first recognition failure at the top level, the system simply suggests that the user "rephrase" the request. After a second failure, the system suggests using "short, simple requests." In usability studies with banking customers in the Fall of 1997, we found that this suggestion typically results in success. Finally, if the user is still unsuccessful, the system suggests the user try specific phrases "like transfer funds from checking to savings or…." The idea is to both suggest specific requests and give examples of the style of request that will likely be successful. In the near future, we plan to quasi-randomly select the example phrases so that the user may be exposed to a number of phrases in the course of a session or over a number of sessions. Progressive prompting is also used within sub-dialogs to guide the user to produce acceptable utterances.

4.2 Dialog manager issues

We designed our dialog managers to be robust with respect to ambiguous parses produced by the semantic grammars and both random and systematic errors made by the speech recognition engine.

For instance, to eliminate confusion resulting from random speech recognition errors, we disallowed nested conversations (e.g. asking for a checking balance in the middle of a money transfer transaction).

It is hard to write a semantic grammar that does not produce an ambiguous parse, even in a limited domain like banking. For example, a user might ask our stock trading system, "*Yes, what is the current volume of IBM*?". This sentence is unambiguous to a human operator. However, a parser might generate two parses: ``[PROCESS = { [ANSWER = {yes}] }, SEM_TYPE = {answer} , SYN_TYPE = {yesno} }], [PROCESS = {[GETINFO = { volume }, SECURITY = {IBM}] }, SEM_TYPE = {info}, SYN_TYPE = {np}]''. Here, each "PROCESS" corresponds to a separate semantic interpretation of the original sentence. The second parse tree is a representation of the volume request, while the first parse tree is a representation of a yes/no response to account for the "Yes" in the beginning of the sentence. We can disambiguate the sentence by mapping the duality of parses above to a volume request using a special semantic signature.

We can consider both ambiguous parse trees and systematic speech recognition errors (e.g. systematic substitution of plurals for singulars) to be special signatures. A dialog manager developer can write special semantic signatures to map these onto unique appropriate actions. Thus we can disambiguate from multiple parses and overcome speech recognition errors by specifically incorporating error conditions into the design of dialog managers.

4.3 Building the NLP component: a lesson in ease of maintenance

As mentioned in the introduction, the conversation machine system has two versions. The version based on construction grammars is currently being deployed, and the version based on semantic grammars, which was developed in parallel, is being tested and will replace the other version in the near future. There were two primary reasons for switching to semantic grammars: the problem of maintaining a construction grammar, and the issue of availability of tools for construction grammar development and maintenance. We realized that supporting and writing new construction grammars would require higher skills than the development group we are working with currently has, and that our research team cannot officially undertake customer support obligations. Therefore we were forced to move towards semantic grammars, where linguistic features are directly mapped onto domain meanings. It is also easier to develop semantic grammar templates for new domains (e.g. stock trading) because tools (e.g., a grammar workbench) are already available to work with semantic grammars.

The moral is that, though it is useful to employ the most advanced natural language understanding technology for a research prototype, the reality of providing system support can force a project towards a simpler, less general, and potentially more costly alternative, which allows the company to move to market faster.

Furthermore, switching to a different technology can open new engineering opportunities. We are currently contemplating the use of machine learning in maintaining and updating semantic grammars. If we are successful, the new ML techniques may also be applicable to similar problems facing construction grammars. This may eventually allow us to move back to using construction grammars for language understanding in commercially viable dialog systems.

4.4 Recovering from speech errors

During the course of building conversation machines for different domains, we used a multi-pronged strategy for handling errors made by the speech recognition module. The accuracy of speech recognition over the telephone at the phoneme, word or even the sentence level is far from perfect, even for the best engines. Thus, a good strategy for error recovery is paramount. Our approach was a multi-pronged one of constraining speech grammars, adding pronunciation models and designing the dialog prompts to enable quick and graceful recovery from errors.

We used the speech grammar reviser (described earlier) to constrain the speech grammar and thereby improve the recognition accuracy by limiting the number of possible sentences that an utterance is potentially interpreted as. For our banking domain, using an initial (over-generalizing) grammar as a baseform, we obtained about a 30-40% increase in recognition accuracy by constraining the grammar to accept only valid domain specific sentences. (Based on anecdotal evidence from 6 people who demonstrate the system to visitors. A more rigorous test is planned for this spring).

We added several pronunciation models for words and phrases to improve recognition accuracy for certain key words or phrases. For instance, for our banking application, it is more important to correctly recognize "checking balance" than "can you tell me" in a sentence "can you tell me my checking balance".

As mentioned earlier, we designed the dialog flow of our conversation machines in such a way that it is easy for a user to quit a transaction in progress, or to access help at any point in the conversation. Also, successive unsuccessful queries by a user result in the playing of more and more helpful messages and eventually after three successive unsuccessful queries, the user is automatically transferred to a service representative.

For transactions where all these techniques (Section 4.4) were used in tandem, we were able to obtain a first try successful transaction rate close to 90%. In a study with 16 participants, 9 women and 7 men, ages 21-50, conducted by an independent agency in fall of 1997. The participants were given a standard set of 15 common phone banking tasks. For different tasks, we obtained first try successful transaction rates ranging from 22% to 90%, proportional to the amount of tuning of the system using the techniques mentioned above (Section 4.4). The *methodology* of the trial is described in (Wolf et al, 1997). In addition, for a demonstration system where the system was tuned extensively using the above techniques, we obtained a first try successful transaction rate close to 90% for *all* tasks based on anecdotal evidence of 6 people who regularly demostrate our system.

Thus, an important lesson here is that even with poor to fair speech recognition accuracy, we can build a very useful and successful system by using techniques like the one described above.

5 Customer issues

Contributing to the challenge of creating a useable system of considerable scope and complexity was the fact that we did not have a complete specification of the functionality to be delivered to the customer. Late in the development cycle, a number of new requirements surfaced, with potentially pervasive effects on the design of the natural language processing and speech grammar components. To avoid the risk of making large changes late in the development cycle, we quickly evaluated several design alternatives with users and picked an acceptable alternative that minimized the need for changes to these components.

Another approach we are taking for managing the scope of the system is to try to anticipate a user's next request based on the current state, and present that anticipated request as an option to the user. This approach avoids resorting to the use of explicit modes, which unnaturally constrain a user's options. For instance, customers often call with the goal of verifying their checking balance. If the balance does not match their records, they frequently ask for checking transaction history in order to determine the cause of the mismatch. When a customer's initial request is for a checking balance, our system asks the user if she wants her checking transaction history, thus reducing the subsequent speech recognition task to a simple yes/no distinction. Similarly, in future systems, when users try to pay a bill but do not have enough money, they will be asked if they want to transfer money and pay the bill. This example is part of an overall strategy of moving towards a system based on user goals as opposed to banking transactions. This goal-based approach reduces recognition errors, and makes the system more efficient for users (see (Wolf and Zadrozny, 1998), for additional examples of design challenges based on customer issues).

Another issue when dealing with real customers and real systems, is the capability of the backend computers to handle complex transactions. For instance, we found that the backend computers of our bank partner were incapable of handling requests for all information relating to a user (as outlined in the above paragraph). Thus, the reality of dealing with primitive backend machines can also constrain an integrated transaction processing system to not being able to process more sophisticated transactions.

6. Conclusions

We have built an integrated AI system for speech enabled transaction processing over the telephone. Our system integrates state of the art technologies from the areas of computer telephony, continuous speech recognition, natural language processing and human computer interaction. Our system differs from other similar systems in the market with respect to both the variety of natural language sentences processed (e.g. the grammar can process around 50 million sentences for our banking system) and the complexity of transactions processed.

Based on the new methodology for research initiatives in our company, we are deploying a prototype "conversation machine" for banking at a customer site. We intend to conduct field trials with actual bank users to continue research on design issues related to dialog flow, computer human interaction, recovery from speech errors and overall integration.

In the process of designing and building conversation machines, the most important lesson we have learnt is that, to build a complete and working system, it is not necessary that each component is perfect. In fact, in most situations, it is impossible to guarantee this. This is true especially in integrated AI systems, where most of the building blocks are error prone. For example, both the continuous speech recognition unit, and, to a lesser extent, the natural language processing unit can produce errors. However, as described earlier, these errors can be overcome to a large extent by careful design and attention to the system modules, the dialog flow and human computer interaction issues, as illustrated by the following examples.

- Our system attempts to predict the user's next query/request and preemptively asks the user if she would like the system to process the query/request. This reduces the task of the speech recognition engine.
- The dialog and prompts in our system guide and gently constrain the user to produce utterances which the system can recognize and understand easily.
- Our system provides feedback about the current transaction at each step of the conversation, to enable users to detect errors, and quickly recover from them.
- Our system progressively prompts the user with increasingly more direction with each consecutive error by suggesting generally successful strategies.

We switched to using semantic grammars for our banking system to enable rapid development and easy maintenance of grammars.

- We add special semantic signatures for common mis-recognitions and ambiguously parsed sentences (for the given domain) to the dialog manager to make it more robust.
- We use a speech grammar reviser tool to constrain the speech grammar and improve recognition accuracy by limiting the number of possible sentences that an utterance is potentially interpreted as.
- We add several pronunciation models for words and phrases which are more important for a given domain topic than other words. For instance, recognition and parsing of the phrase "checking balance" is paramount to successfully handling any sentence which includes this phrase for our banking system.

For the transactions for which we used all the above techniques in tandem, we observed that users of our telephone banking system obtained a successful transaction rate close to 90%.

In future, we intend to work on developing machine learning algorithms for automatically mapping semantic signatures to domain topics, conduct research on allowing nested dialogs in applications and build Conversation Machines for larger domains.

We conclude that, with our new research methodology, it is possible to build a very robust and useful natural language dialog system, which integrates the AI technologies of telephone speech recognition, natural language processing and human computer interaction. We can overcome deficiencies in any of the component modules by paying careful attention to the dialog flow and mechanisms for error recovery as outlined in this paper. In the process, having a customer enables us to make more realistic design choices.

References

C.J. Fillmore, P. Kay, and M.C. O'Connor. (1988). Regularity and idiomaticity in grammatical constructions. *Language*, 64(3):501--538.

G. Gazdar and C.S. Mellish (1989). Natural language processing in Prolog: an introduction to computational linguistics. Addison Wesley.

A.E. Goldberg. (1995). *Constructions: a construction grammar approach to argument structure.* The University of Chicago Press, Chicago, IL.

D. Jurafsky. (1992). An On-line Computational Model of Sentence Interpretation. *PhD thesis, University of California, Berkeley*, Report No. UCB/CSD 92/676.

N. Kambhatla and W. Zadrozny (1998). Automated Grammar Revision Using Counter-Examples. *IBM T.J. Watson Research Center* Technical Report.

Oviatt, S. (1995). Predicting spoken disfluencies during human-computer interaction. *Computer Speech and Language, 9*, 19-35.

Wolf, C. G., Kassler, M., Zadrozny, W., Opyrchal, L. (1997). *Interact 97 Conference Proceedings*, 461-468.

Wolf, C. G. and Zadrozny, W. (1998). Evolution of the Conversation Machine: A case study of bringing advanced technology to the marketplace. *CHI'98 Conference Proceedings*, to appear.

Yankelovich, N. (1996). How do users know what to say? *Interactions*, 3.6, 32-43.

Yankelovich, N., Levow, G. A., and Marx, M. (1995). Designing SpeechActs: Issues in speech user interfaces. *CHI '95 Conference Proceedings*, 369-376.