PartsID: A Dialogue-Based System for Identifying Parts for Medical Systems

Amit BAGGA, Tomek STRZALKOWSKI, and G. Bowden WISE
Information Technology Laboratory
GE Corporate Research and Development
1 Research Circle
Niskayuna, USA, NY 12309
{bagga, strzalkowski, wisegb}@crd.ge.com

Abstract

This paper describes a system that provides customer service by allowing users to retrieve identification numbers of parts for medical systems using spoken natural language dialogue. The paper also presents an evaluation of the system which shows that the system successfully retrieves the identification numbers of approximately 80% of the parts.

Introduction

Currently people deal with customer service centers either over the phone or on the world wide web on a regular basis. These service centers support a wide variety of tasks including checking the balance of a bank or a credit card account, transferring money from one account to another, buying airline tickets, and filing one's income tax returns. Most of these customer service centers use interactive voice response (IVR) systems on the front-end for determining the user's need by providing a list of options that the user can choose from, and then routing the call appropriately. The IVRs also gather essential information like the user's bank account number, social security number, etc. For back-end support, the customer service centers use either specialized computer systems (example: a system that retrieves the account balance from a database), or, as in most cases, human operators.

However, the IVR systems are unwieldy to use. Often a user's needs are not covered by the options provided by the system forcing the user to hit 0 to transfer to a human operator. In addition, frequent users often memorize the sequence of options that will get them the desired information. Therefore, any change in the options greatly inconveniences these users. Moreover, there are users that always hit 0 to speak to a live operator because they prefer to deal with a human instead of a machine. Finally, as customer service providers continue to rapidly add functionality to their IVR systems, the size and complexity of these systems continues to grow proportionally. In some popular systems like the IVR system that provides customer service for the Internal Revenue Service (IRS), the user is initially bombarded with 10 different options with each option leading to sub-menus offering a further 3-5 options, and so on. The total number of nodes in the tree corresponding to the IRS' IVR system is quite large (approximately 100) making it extremely complex to use.

Some customer service providers have started to take advantage of the recent advances in speech recognition technology. Therefore, some of the IVR systems now allow users to say the option number (1, 2, 3, ..., etc.) instead of pressing the corresponding button. In addition, some providers have taken this a step further by allowing users to say a keyword or a phrase from a list of keywords and/or phrases. For example, AT&T, the long distance company, provides their users the following options: "Please say information for information on placing a call, credit for requesting credit, or operator to speak to an operator."

However, given the improved speech recognition technology, and the research done in natural language dialogue over the last decade, there exists tremendous potential in enhancing

these customer service centers by allowing users to conduct a more natural human-like dialogue with an automated system to provide a customer-friendly system. In this paper we describe a system that uses natural language dialogue to provide customer service for a medical domain. The system allows field engineers to call and obtain identification numbers of parts for medical systems using natural language dialogue. We first describe some work done previously in using natural language dialogue for customer service applications. Next, we present the architecture of our system along with a description of each of the key components. Finally, we conclude by providing results from an evaluation of the system.

1. Previous Work

As mentioned earlier, some customer service centers now allow users to say either the option number or a keyword from a list of options/descriptions. However, the only known work which automates part of a customer service center using natural language dialogue is the one by Chu-Carroll and Carpenter (1999). The system described here is used as the front-end of a bank's customer service center. It routes calls by extracting key phrases from a user utterance and then by statistically comparing these phrases to phrases extracted from utterances in a training corpus consisting of pre-recorded calls where the routing was done by a human. The call is routed to the destination of the utterance from the training corpus that is most "similar" to the current utterance. On occasion, the system will interact with the user to clarify the user's request by asking a question. For example, if the user wishes to reach the loan department, the system will ask if the loan is for an automobile, or a home. Other related work is (Georgila et al., 1998).

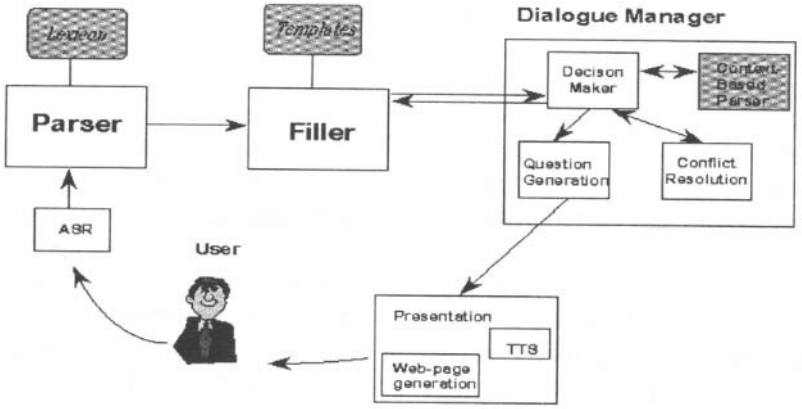
While we are aware of the work being done by speech recognition companies like Nuance (www.nuance.com) and Speechworks (www.speechworks.com) in the area of providing more natural language dialogue-based customer service, we are not aware of any conference or journal publications from them. Some magazine articles which mention their work are (Rosen 1999 ; Rossheim 1999 ; Greenemeier 1999 ; Meisel 1999). In addition, when we tried out a demo of Nuance's systems, we found that their systems had a very IVRish feel to them. For example, if one wanted to transfer $50 from one account to another, the system would first ask the account that the money was coming from, then the account that the money was going to, and finally, the amount to be transferred. Therefore, a user could not say "I want to transfer $50 from my savings account to my checking account" and have the system conduct that transaction.

In addition to the works mentioned above, there have been several classic projects in the area of natural language dialogue like TRAINS/TRIPS project at Rochester (Allen et al., 1989, 1995, 1996), Duke's Circuit-Fixit-Shoppe and Pascal Tutoring System (Biermann et al., 1997; 1995), etc. While the Circuit-Fixit-Shoppe system helps users fix a circuit through a dialogue with the system, the TRIPS and the TRAINS projects allow users to plan their itineraries through dialogue. Duke's Pascal tutoring system helps students in an introductory programming class debug their programs by allowing them to analyze their syntax errors, get additional information on the error, and learn the correct syntax. Although these systems have been quite successful, they use detailed models of the domain and therefore cannot be used for diverse applications such as the ones required for customer service centers. Other related work on dialogue include (Carberry, 1990; Grosz and Sidner, 1986; Reichman, 1981).

2. PartsID: A System for Identification of Parts for Medical Systems

Initially, we were approached by the medical systems business of our company for help in reducing the number of calls handled by human operators at their call center. An analysis of the types of customer service provided by their call center showed that a large volume of calls handled by their operators were placed by field engineers requesting identification numbers of parts for various medical systems. The ID numbers were most often used for ordering the corresponding parts using an automated IVR system. Therefore, the system we have built

Figure 1. PartsID System Architecture



helps automate some percentage of these calls by allowing the engineer to describe a part using natural language. The rest of this section describes our system in detail.

2.1 Data

The database we used for our system was the same as the one used by the operators at the call center. This database consists of the most common parts and was built by the operators themselves. However, the data contained in the database is not clean and there are several types of errors including mis-spellings, use of non-standard abbreviations, use of several different abbreviations for the same word, etc.

The database consists of approximately 7000 different parts. For each part, the database contains its identification number, a description, and the product (machine type) that it is used in. The descriptions consist of approximately 60,000 unique words of which approximately 3,000 are words which either are non-standard abbreviations or are unique to the medical domain (example: collimator).

Due to the large size of the database, we did not attempt to clean the data. However, we did build several data structures based on the database which were used by the system. The primary data structures built were two inverted hash tables corresponding to the product, and the part description fields in the database. The inverted hash tables were built as follows:

1) Each product and part description field was split into words.
2) Stop-words (words containing no information like: a, the, an, etc.) were filtered.
3) Each remaining word was inserted as the index of the appropriate hash table with the identification number of the part being the value corresponding to the index.

Therefore, for each non-stop-word word used in describing a part, the hash table contains a list of all the parts whose descriptions contained that word. Similarly, the products hash table contains a list of all parts corresponding to each product word.

2.2 System Architecture

The architecture of the system is shown in Figure 1. The system was designed in a manner such that it could be easily ported from one application to another with minimal effort other than providing the domain-specific knowledge regarding the new application. Therefore, we decided to abstract away the domain-specific information into self-contained modules while keeping the other modules completely independent. The domain-specific modules are shown in the dark shaded boxes in Figure 1. The remainder of this section discusses each of the modules shown in the system architecture.

2.2.1 The Speech Recognition System (ASR)

Since customer service centers are meant to be used by a variety of users, we needed a user-independent speech recognition system. In

addition, since the system could not restrict the manner in which a user asked for service, the speech recognition system could not be grammar-based. Therefore, we used a general purpose dictation engine for the system. The dictation system used was Lernout & Hauspie's VoiceXPress system (www.lhs.com). Although the system was general purpose, we did provide to it the set of keywords and phrases that are commonly used in the domain thereby enabling it to better recognize these domain-specific keywords and phrases. The keywords and phrases used were simply the list of descriptions and product names corresponding to each part in the database. It should be noted that the set of domain-specific keywords and phrases was provided to the speech recognition system as a text document. In other words, the training was not done by a human speaking the keywords and phrases into the speech recognition system. In addition, the speech recognition system is far from perfect. The recognition rates hover around 50%, and the system has additional difficulty in identifying product names which are most often words not found in a dictionary (examples: 3MlaserCam, 8000BUCKY, etc.).

2.2.2 Parser and the Lexicon

The parser is domain-driven in the sense that it uses domain-dependent information produced by the lexicon to look for information, in a user utterance, that is useful in the current domain. However, it does not attempt to understand fully each user utterance. It is robust enough to handle ungrammatical sentences, short phrases, and sentences that contain mis-recognized text.

The lexicon, in addition to providing domain-dependent keywords and phrases to the parser, also provides the semantic knowledge associated with each keyword and phrase. Therefore, for each content word in the inverted hash tables, the lexicon contains entries which help the system determine whether the word was used in a part description, or a product name. In addition, the lexicon also provides the semantic knowledge associated with the pre-specified actions which can be taken by the user like "*operator*" which allows the user to transfer to an operator, and "*stop*," or "*quit*" which allow the user to quit the system. Some sample entries are:

collimator => (description_word,collimator)
camera => (product_word, camera)
operator => (user action, operator)
etc.

The parser scans a user utterance and returns, as output, a list of semantic tuples associated with each keyword/phrase contained in the utterance. It is mainly interested in "key words" (words that are contained in product and part descriptions, user action words, etc.) and it ignores all the other words in the user utterance. The parser also returns a special tuple containing the entire input string which may be used later by the context-based parser for sub-string matching specially in cases when the DM has asked a specific question to the user and is expecting a particular kind of response.

2.2.3 The Filler and Template Modules

The filler takes as input the set of tuples generated by the parser and attempts to check off templates contained in the templates module using these tuples. The set of templates in the templates module contains most of remaining domain-specific knowledge required by the system. Each template is an internal representation of a part in the database. It contains for each part, its ID, its description, and the product which contains it. In addition, there are several additional templates corresponding to pre-specified user actions like "*operator*," and "*quit*." A sample template follows:

t1_1 = (
'product' => 'SFD',
'product_ids' => '2229005',
'product_descriptions' => 'IR RECEIVER PC BOARD CI104 BISTABLE MEMORY')

For each tuple input from the parser, the filler checks off the fields which correspond to the tuple. For example, if the filler gets as input *(description_word, collimator)*, it checks off the description fields of those templates containing *collimator* as a word in the field. A template is checked off *iff* one or more of its fields is checked off. In addition, the filler also maintains a list of all description and product words passed through the tuples (i.e. these words

have been uttered by the user). These two lists are subsequently passed to the dialogue manager.

Although the filler does not appear to be very helpful for the current application domain, it is an important part of the architecture for other application domains. For example, the current PartsID system is a descendant from an earlier system which allowed users to process financial transactions where the filler was instrumental in helping the dialogue manager determine the type of transaction being carried out by the user (Bagga et al., 2000).

2.2.4 The Dialogue Manager (DM)

The DM receives as input from the filler the set of templates which are checked off. In addition, it also receives two lists containing the list of description words, and product word uttered by the user. The DM proceeds using the following algorithm:

1) It first checks the set of checked off templates input from the filler. If there is exactly one template in this set, the DM asks the user to confirm the part that the template corresponds to. Upon receipt of the confirmation from the user, it returns the identification number of the part to the user.
2) Otherwise, for each description word uttered by the user, the DM looks up the set of parts (or templates) containing the word from the descriptions inverted hash table. It then computes the intersection of these sets. If the intersection is empty, the DM computes the union of these sets and proceeds treating the union as the intersection.
3) If the intersection obtained from (2) above contains exactly one template, the DM asks the user to confirm the part corresponding to the template as in (1) above.
4) Otherwise, the DM looks at the set of product words uttered by the user. If this set is empty, the DM queries the user for the product name. Since the DM is expecting a product name here, the input provided by the user is handled by the context-based parser. Since most product names consist of non-standard words consisting of alpha-numeric characters (examples: AMX3, 8000BUCKY, etc.), the recognition quality is quite poor. Therefore, the context-based parser ranks the input received from the user using a sub-string matching algorithm that uses character-based unigram and bigram counts (details are provided in the next section). The sub-string matching algorithm greatly enhances the performance of the system (as shown in the sample dialogue below).
5) If the set of product words is non-empty, or if the DM has successfully queried the user for a product name, it extracts the set of parts (templates) containing each product word from the product words inverted hash table. It then computes an intersection of these sets with the intersection set of description words obtained from (2) above. The resulting intersection is the joint product and description intersection.
6) If the joint intersection has exactly one template, the DM proceeds as in (1) above. Alternatively, if the number of templates in the joint intersection is less than 4, the DM lists the parts corresponding to each of these and asks the user to confirm the correct one.
7) If there are more than 4 templates in the joint intersection, the DM ranks the templates based upon word overlap with the description words uttered by the user. If the number of resulting top-ranked templates is less than 4, the DM proceeds as in the second half of (6) above.
8) If the joint intersection is empty, or in the highly unlikely case of there being more than 4 top-ranked templates in (7), the DM asks the user to enter additional disambiguating information.

The goal of the DM is to hone in on the part (template) desired by the user, and it has to determine this from the set of templates input to it by the filler. It has to be robust enough to deal with poor recognition quality, inadequate information input by the user, and ambiguous data. Therefore, the DM is designed to handle these issues. For example, description words that are mis-recognized as other description words usually cause the intersection of the sets of parts corresponding to these words to be empty. The DM, in this case, takes a union of the sets of parts corresponding to the description

words thereby ensuring that the template corresponding to the desired part is in the union.

The DM navigates the space of possibilities by first analyzing the intersection of the sets of parts corresponding to the description words uttered by the user. If no unique part emerges, the DM then checks to see if the user has provided any information about the product that the part is going to be used in. If no product was mentioned by the user, the DM queries the user for the product name. Once this is obtained, the DM then checks to see if a unique part corresponds to the product name and the part description provided by the user. If no unique part emerges, then the DM backs off and asks the user to re-enter the part description. Alternatively, if more than one part corresponds to the specified product and part description, then the DM ranks the parts based upon the number of words uttered by the user. Obviously, since the DM in this case uses a heuristic, it asks the user to confirm the part that ranks the highest. If more than one (although less than 4) parts have the same rank, then the DM explicitly lists these parts and asks the user to specify the desired part. It should be noted that the DM has to ensure that the information it receives is actually what the user meant. This is especially true when the DM uses heuristics, and sub-string matches (as in the case of product names). Therefore, the DM occasionally asks the user to confirm input it has received.

2.2.5 The Sub-String Matching Algorithm

When the dialogue manager is expecting a certain type of input (examples : product names, yes/no responses) from the user, the user response is processed by the context-based parser. Since the type of input is known, the context-based parser uses a sub-string matching algorithm that uses character-based unigram and bigram counts to match the user input with the expectation of the dialogue manager. Therefore, the sub-string matching module takes as input a user utterance string along with a list of expected responses, and it ranks the list of expected responses based upon the user response. Listed below are the details of the algorithm :

1) The algorithm first concatenates the words of the user utterance into one long string. This is needed because the speech recognition system often breaks up the utterance into words even though a single word is being said. For example, the product name AMX110 is often broken up into the string *'Amex 110'*.
2) Next, the algorithm goes through the string formed in (1) and compares this character by character with the list of expected responses. It assigns one point for every common character. Therefore, the expected response 'AMX3' gets three points for the utterance 'Amex110'.
3) The algorithm then compares the user utterance with the list of expected responses using 2 characters (bigrams) at a time. It assigns 2 points for each bigram match. For the example shown in (2), there are two bigram matches : the first is that the utterance starts with an 'A' (the previous character is this case is the null character), and the second is the bigram 'AM'.
4) The algorithm now compares the length of the user utterance string and the expected response. If the length of the two strings is the same, then it assigns 2 points to the expected response.
5) Finally, the algorithm calculates the number of unique characters in the expected response, and the user utterance string. If these characters are the same, then it assigns 4 points to the expected response.

The expected response which has the highest number of points is the most likely one. If two or more expected responses have the same number of points, then the system asks the user to confirm the correct one.

While we have not evaluated this sub-string matching algorithm independently, a brief evaluation in the context of the system resulted in about 90% accuracy.

2.2.6 The Presentation Module

The presentation module works in one of two possible modes: over the phone, and over the web. This module takes as input a string generated by the question-generation module and presents this string to the user in the appropriate mode of communication. If the speech option for the system is turned on, the speech-based output is generated using Lernout

and Hauspie's RealSpeak text-to-speech system. Although the system currently cannot use both modes of communication simultaneously, we plan to incorporate this feature sometime in the future.

2.2.7 Robustness of the System

As with any dialogue system, it is extremely important for the system to be robust. Our system has the following two features which make it extremely robust:

1) The user can, at any given moment, say *operator* to transfer to an operator, *quit/stop* to exit, and *back/restart* to start afresh.
2) When expecting a response from the user, if the system does not receive an expected input, it repeats the question at most twice before transferring control to an operator.

3. Sample Dialogues

This section gives examples of two sample dialogues that occur during the testing of the system. The system's response is indicated by "S>", the user's response by "U>", and the recognition of the user's response by "V>".

S> Welcome to the PartsID Center. Please tell me the part that you want the ID for.
U> I need the ID for PCB Control.
V> I need the I the for PCB home thrown.

S> What product do you need this for?
U> COMPAX 40E
V> home fax for a 00 E

S> Is COMPAX 40E the product name that you said?
U> yes
V> gas

S> Do you want the ID for PCB Display SID or the PCB Control?
U> The second one.
V> The second home.

S> Do you need the ID for PCB Control used in product COMPAX 40E?
U> yes
V> yes

S> The ID for PCB Control used in product COMPAX 40E is 2168983. Do you wan to exit or find the ID for another part?
U> look up another part.

S> Welcome to the PartsID Center. Please tell me the part that you want the ID for.
U> I need cable data for cradle 2M long.
V> I needed cable date them for cradle to be loan.

S> Do you need this part for a ANGIXM2004, or a AFM?
U> AFM
V> a fan

S> Is AFM the product name that you said?
U> yes
V> yes

S> Do you want the ID for Cable Data for Cradle 2M long?
U> yes
V> yes

S> The ID for Cable Data for Cradle 2M long used in product AFM is 45438628. Do you wan to exit or find the ID for another part?
U> look up another part.

4. Evaluation and Results

The goal of our evaluation was to ensure that the system helped a user successfully identify parts irrespective of the performance of the speech recognition engine for the user. In other words, we wanted to see if the system was robust enough to conduct transactions with a diverse mix of users. We tested the system with 4 different users two of whom had foreign accents. For each user, we randomly selected 20 parts from the database. The results are summarized in Table 1.

These results show that the system was quite successful in handling requests from users with a variety of accents achieving varying recognition rates. Out of the 80 parts tested, only twice did the user feel that he/she had to transfer to an operator. The system successfully retrieved the identification numbers of 79% of the parts while transferring 19% of the cases to a human operator because of extremely bad

User	Parts successfully identified	Calls system transfers to operator	Calls user transfers to operator	System prompts per call	Relevant words recognized per part
1	15	3	2	3.7	2.5
2	18	2	0	3	2.35
3	13	7	0	2.5	1.65
4	17	3	0	2.9	2.7

Table 1: Summary of Results

recognition. We are planning on conducting a more elaborate test which a larger set of users.

Conclusions

In this paper we have described a robust system that provides customer service for a medical parts application. The preliminary results are extremely encouraging with the system being able to successfully process approximately 80% of the requests from users with diverse accents.

Acknowledgements

We wish to thank the GE Medical Systems team of Todd Reinke, Jim Tierney, and Lisa Naughton for providing support and funding for this project. In addition, we also wish to thank Dong Hsu of Lernout and Hauspie for his help on the ASR and the text-to-speech systems. Finally, we wish to thank the Information Technology Laboratory of GE CRD for providing additional funding for this project.

References

Allen, J. F. et al. (1995) *The TRAINS Project: A case study in building a conversational planning agent.* Journal of Experimental and Theoretical AI, (7) 7-48.

Allen, J. F., Miller, B. W.; Ringer, E. K.; and Sikorski, T. (1996) *A Robust System for Natural Spoken Dialogue.* 34th Annual Meeting of the ACL, Santa Cruz, 62-70.

Bagga, A., Stein G. C., and Strzalkowski, T. (2000) *FidelityXPress: A Multi-Modal System for Financial Transactions.* Proceedings of the 6th Conference on Content-Based Multimedia Information Access (RIAO'00).

Biermann, A.W.; Rodman, R.; Rubin, D.; and Heidlage, J.R. (1985) *Natural language with discrete speech as a mode for human to machine communication.* Communication of the ACM 18(6): 628-636.

Biermann, Alan W.; Guinn, Curry I.; Fulkerson, M.: Keim, G.A.; Liang, Z.; Melamed, D.M.; and Rajagopalan, K. (1997) *Goal-oriented Multimedia Dialogue with Variable Initiative.* Lecture Notes in Artificial Intelligence 1325; Springer-Verlag, New York; pp. 1-16.

Carberry, S. (1990) *Plan Recognition in Natural Language Dialogue.* Cambridge, Mass.: The MIT Press.

Chu-Carroll, J, and R. Carpenter. (1999) *Vector-Based Natural Language Call Routing.* Journal of Computational Linguistics, 25(30), pp. 361-388.

Georgila, K., A.Tsopanoglou, N.Fakotakis and G.Kokkinakis. (1998) *An Integrated Dialogue System for the Automation of Call Centre Services.* ICLSP'98, 5th International Conference on Spoken Language Processing, Sydney, Australia.

Grosz, B.J. and Sidner, C.L. (1986) *Attentions, intentions, and the structure of discourse.* Computational Linguistics 12(3): 175-204.

Greenemeier, L. (1999) *Voice-Recognition Technology Builds a Following.* Information Week, December 13.

Meisel, W. (1999) *Can Speech Recognition Give Telephones a New Face?* Business Communications Review, November 1.

Reichman, R.. (1981) Plain-speaking: *A theory and grammar of spontaneous discourse.* PhD thesis, Department of Computer Science, Harvard University, Cambridge, Massachusetts.

Rosen, C. (1999) Speech Has Industry Talking. Business Travel News, November.

Rossheim, J. (1999) *Giving Voice to Customer Service.* Datamation, November 1.