Towards the Automatic Generation of Mixed-Initiative Dialogue Systems from Web Content[1]

Joseph Polifroni†, Grace Chung‡, and Stephanie Seneff†

†Spoken Language Systems Group
MIT Laboratory for Computer Science
Cambridge, Massachusetts USA
{joe,seneff}@sls.lcs.mit.edu

‡Corporation for National Research Initiatives
Reston, Virginia, USA
gchung@cnri.reston.va.us

Abstract

Through efforts over the past fifteen years, we have acquired a great deal of experience in designing spoken dialogue systems that provide access to large corpora of data in a variety of different knowledge domains, such as flights, hotels, restaurants, weather, etc. In our recent research, we have begun to shift our focus towards developing tools that enable the rapid development of new applications. This paper addresses a novel approach that drives system design from the on-line knowledge resource. We were motivated by a desire to minimize the need for a pre-determined dialogue flow. In our approach, decisions on dialogue flow are made dynamically based on analyses of data, either prior to user interaction or during the dialogue itself. Automated methods, used to organize numeric and symbolic data, can be applied at every turn, as user constraints are being specified. This helps the user mine through large data sets to a few choices by allowing the system to synthesize intelligent summaries of the data, created on-the-fly at every turn. Moreover automatic methods are ultimately more robust against the frequent changes to on-line content. Simulations generating hundreds of dialogues have produced log files that allow us to assess and improve system behavior, including system responses and interactions with the dialogue flow. Together, these techniques are aimed towards the goal of instantiating new domains with little or no input from a human developer.

1. Introduction

Over the past fifteen years, researchers in the Spoken Language Systems Group at MIT have been developing human language technologies for *mixed-initiative* conversational systems that help humans seek information from resources such as the Web. These are distinguished from the emerging deployed commercial systems in that the interaction is natural and flexible, modeled after human-human dialogues [8]. The development of the Galaxy Communicator architecture [6] has greatly accelerated the pace at which we as experts can configure complex dialogue systems in a wide range of different domains. As the underlying technology components have matured, the community's research focus has evolved to include issues related to portability and modularity of system components [7, 2]. The goal is to automate the process of both customizing the system to specific content and designing the dialogue flow. At the same time, the system should be robust and adaptable to the frequent changes in the on-line content. We believe that the ability for non-expert system developers to rapidly configure spoken dialogue access to on-line information sources will be a crucial step towards wide deployment of such systems.

Our current vision is that the system development process should begin with the on-line resource. The first step is to transform it into a structured database, while simultaneously identifying the relevant attributes that would lead to successful retrieval of an item from the database. We assume, as a reasonable model of a dialogue, a sequence of dialogue turns, aimed at narrowing down the full database to a single item. The item retrieved will match the set of constraints provided by the user over the course of the dialogue. While the user may choose to provide these constraints in any order, they will be strongly biased by the information presented to them at each turn. It is thus required of the system to intelligently present a summary of the set of database tuples retrieved at each turn, and, optionally, to suggest a plausible next move to the user, aimed at further refinement of the set. Towards this goal, we have developed a capability which can effectively organize a list of database tuples into a summary semantic frame. The frame is then transformed into a response string using our language generation tools.

This paper begins with an outline of past work in domain-independent dialogue management. We then describe progress in formulating a procedure where knowledge is extracted from on-line data prior to user interaction, and then used to dynamically configure dialogue flow and summaries of database outputs, based on the dialogue contexts. We give examples of outputs from our hotel and restaurant information domains. Section 5 addresses how the language generation engine is used to create clear and concise responses from the dynamically synthesized summary frames. Subsequently, we present a methodology for creating and fine-tuning the restaurant domain. In particular a simulation server is used to randomly generate hundreds of dialogues, towards the goal of refining all aspects of the system. Finally, we discuss future directions towards the automatic generation of dialogue systems.

2. Generic Dialogue Management

A previous paper [4] describes initial efforts to build a domain-independent or *generic* dialogue manager that can perform the essential dialogue flow operations, customizable through an external, text-based interface. Our philosophy is that the basic functionalities in the dialogue manager should obtain their domain-dependent parameters from external files, separate from

[1] The research at MIT was supported by NTT, as well as by an industrial consortium supporting the Spoken Language Systems Group. The research at CNRI was supported by DARPA under contract number N66001-00-2-8922, monitored through SPAWAR Systems Center, San Diego.

	Cluster 1	Cluster 2	Cluster 3
City	Range (N)	Range (N)	Range (N)
Hotel Rates			
NYC	65-210 (127)	210-475 (103)	475-550 (15)
BOS	59-139 (58)	139-415 (71)	415-599 (9)
PHL	49-129 (69)	129-295 (51)	295-390 (6)
SUX	33-90 (15)	99 (1)	n/a
Distance from City Center			
NYC	0-1 (207)	1-11 (24)	11-13 (5)
BOS	0-20 (105)	20-45 (27)	45-55 (3)
PHL	0-26 (107)	26-30 (8)	30-40 (7)
SUX	0-14 (3)	14-15 (2)	15-201 (1)

Table 1: Results of clustering of hotel nightly rates and distance from city center for selected cities: New York (NYC), Boston (BOS), Philadelphia (PHL) and Sioux City (SUX). Range is the output range of values in dollars for price and miles for distance; N represents number of items in each grouping.

the domain-independent components. For instance, an application has a set of domain-specific constraints, such as "arrival date" in a flight domain. These need to be solicited from the user in order to find the required database item.

The design of this generic dialogue manager is based on the view that the dialogue interaction consists of several phases. In the *pre-retrieval* phase, the system solicits from the user *all* the necessary constraints prior to dispatching a query with those constraints to the content provider. Subsequent phases involve further filtering the response as well as preparing a summary of the retrieved data set to the user. This model mandates a developer to pre-determine a fixed, ordered set of constraints to prompt a user for each dialogue turn, if necessary. In recent work, we have adopted a simplified and more flexible paradigm where database retrieval occurs as early as possible, following which, the system will examine the large database output, provide a summary of the data, and prompt the user for further constraints, based on the data. This approach is ultimately more data-centric, resulting in dialogues that are more natural.

3. Extraction of On-line Content

A significant challenge in building spoken dialogue applications has been the representation and organization of the knowledge source. Typically, applications in narrow domains are designed to connect to a single set of database tuples containing a finite and fixed set of attributes that apply exclusively to that domain. Assumptions based on one corpus can result in a system that is difficult to port to even a new corpus of data within the same domain, much less to a completely different domain. For instance, data sets for a restaurant domain can vary greatly in size or specificity for different cities. A database of restaurants in New York is much more diverse than one for a small town. Only a small subset of the available cities would provide access to restaurant reviews, and the concept of "neighborhoods" is only applicable to cities over a certain size. Using a static set of attributes, incorrect assumptions could be made about new data, or in the worst case, a system needs to be substantially re-engineered every time a new data set is adopted.

The solution we propose is to make dynamic decisions on dialogue flow based on an analysis of the data, either prior to user interaction or during the dialogue itself. In our initial data analysis phase, content is organized into subcategories, corresponding to the high-level concepts of the domain as determined by the provider. For example, with hotel data, the initial pass categorizes data into hotel names, cities, addresses, etc. The data are further standardized into semantic frames by way of parsing using a natural language engine [5]. Capturing knowledge extracted from the raw data, lists of semantic frames can be compiled under several subcategories, such as all hotels in a given city. These subcategories can be computed prior to user interaction and stored as a data resource (e.g., lists of database entities for a given city). In addition, on-the-fly computation takes place after a user has specified further constraints. Throughout the dialogue, the system examines the subset of frames and the characteristics of each *novel* data subset in the hope of giving more informative and pertinent feedback to the user at every turn.

4. Organization and Presentation

In this section, we first describe how *numeric* values can be automatically organized into groupings that could map to subjective categories, such as "cheap" or "near." Secondly, an algorithm for automatically organizing *symbolic* data is presented. Our overall goal is to fully automate the process of deciding how to summarize database entries that are consistent with the user's request.

These ideas are explored in the context of two data sets: (1) restaurants in Boston and (2) hotel data for major cities in the United States. The hotel content includes information about brand identification, location, and amenities offered, as well as numeric ranges for minimum and maximum price, and the distance (in miles or kilometers) from a specified landmark (the default being the center of the city in question). The hotel information is accessible via a direct connection to a provider, whereas the restaurant content was obtained by processing an on-line source. It has information for 983 restaurants, covering 106 cities in the Boston metropolitan area (e.g., Newton, Cambridge) and 45 neighborhoods (e.g., Back Bay, South End). Altogether there are 33 different cuisines, as well as information about phone number, hours, credit-card acceptance, handicap accessibility, available reviews, etc. The on-line source for restaurants provides price range information in four specific categories, "cheap," "low," "mid-range" and "expensive," whereas the hotel information provides numeric ranges for hotel prices.

4.1. Grouping Numeric Data

In earlier work [4] within the hotel information domain, partitioning of the data was based on a fixed numeric value for a given attribute, supplied as a parameter in a domain-independent function. Hence, to determine if a hotel is "near" a landmark, a developer specified a fixed threshold such as 10 miles as a criterion for "nearness." This is clearly inadequate in a domain that covers a variety of American cities, in that one fixed numeric value cannot match our intuitions for relative concepts such as "near" and "cheap" for entire ranges of cities. It would be equally impractical for a system developer to incorporate varying notions for "near"/"cheap" for each city, suburb or neighborhood manually. This clearly argues for a more automatic method that can infer relative concepts from the data source itself.

Our approach is to use an algorithm, similar to bottom-up clustering, whereby a numerically ordered data set is sorted into bins that roughly correspond with notions of "small," "medium," and "large." The algorithm begins with one bin allo-

Restaurants in Quincy: "I have found 14 restaurants. Some of the options are American, Brazilian, Indian and Italian Pizza. None of them are expensive. Many of them are on Hancock Street, Adams Street, Billings Road and Franklin Street."
Restaurants with a medium price range: "I have found 172 restaurants. Most of them are located in Boston and Cambridge. There are 18 choices for cuisine. They are predominantly in the North End, Back Bay, the Financial District and the South End."
Restaurants near the Prudential Center: "I have found 21 restaurants. There are 11 choices for cuisine. Many of them are on Boylston Street, Newbury Street and Massachusetts Avenue."

Table 2: Example summary responses for various sets of restaurants. The first one corresponds to the response frame in Table 3.

cated to every unique value in the set, and successively merges adjacent bins whenever the distance between the means of the bin falls below a varying threshold. The results of this algorithm applied to hotel data are shown in Table 1. We illustrate results for the price attribute and distance attribute from the city center. The results highlight a difference in the notion of what is cheap for New York compared with Sioux City. Likewise, when this algorithm is applied to data pertaining to distances from the city center, the notion of "near" in Manhattan maps to within 1 mile for over 200 hotels, which contrasts significantly with the data spread in a city such as Philadelphia.

4.2. Organizing Symbolic Data

Just as numeric data typically have different ranges and sizes in parts of the corpus, the distributions of *symbolic* values also vary among subsets, or may be absent altogether. As described earlier, each dialogue turn results in a list of semantic frames that encapsulate knowledge extracted from raw data, corresponding to constraints specified in the dialogue. The system prompts the user with attributes such as city or cuisine, in order to solicit more constraints, aimed at narrowing the subset of frames, although users are always free to ignore these suggestions. Previously, both the set of constraints and the order by which to prompt for them was pre-determined by a developer.

In our new paradigm, we abandon the notion of a predetermined dialogue flow. The prompts presented to the user, and the order in which they appear, are now determined at runtime, based on a simple algorithm which computes the most useful set of attributes, as dictated by the *current* data subset. The algorithm analyzes these data and produces a frame that summarizes the attributes associated with them. The response generation component will then use the attributes as well as other details in the frames to generate a response that both summarizes the current data and suggests further constraints.

The algorithm is implemented in an independent server that generates a set of attributes at every turn. Certain attributes, such as phone or street numbers, may be excluded *a priori* by a developer as inappropriate for use to constrain a query. With the remainder of the attributes, the algorithm will, at each turn, select a subset that seem most relevant to the user, given the current data set. The list of candidate attributes in the restaurant domain include city, neighborhood, price range, restaurant ratings and street location.

The decision on the usefulness of an attribute is based on two main considerations. First, for a large data set, attributes that partition the data into a small set of unique groups are more useful than those that are mostly unique across all the items. The rationale is that the system should at first summarize at a broad level, presenting the user with a set of choices to narrow down the data. For instance, at an earlier part of the conversation, the system may decide that it is preferable to prompt the user for a cuisine choice rather than asking for a specific street location. Secondly, attributes for which the majority of the tuples have no values are also removed from consideration. For instance, in some suburbs none of the listed restaurants have ratings.

Results of this summarization algorithm are passed on to the response generation component in the form of summary frames, as shown in Table 3. Each summary frame contains a set of attributes, each one containing a frequency-ordered set of values for that attribute, along with their counts. The next section describes how these are converted to English sentences.

5. System Responses

Once the system is able to produce a reasonable summary meaning representation for the data, the next step is to turn that into a sequence of well-formed English sentences to be spoken to the user as the system's verbal response. For this, we make use of our GENESIS language generation system [1]. GENESIS is mostly a surface-form generator, but it allows for a certain amount of planning, so as to generate a particular item differently depending upon further analysis of its contents or the external context.

Some example summary generations are given in Table 2, where the first one corresponds to the frame in Table 3. GENESIS provides flexible options to list the few dominating choices, if they exist, or to list all the choices, if the set is sufficiently small, or to simply indicate how many choices there are. There is also the option to only provide detail if there is a single choice, or, interestingly, to summarize across all tuples for attributes that are either common or missing (e.g., "none of them is expensive").

GENESIS has been augmented to accommodate a number of special operators for preprocessing summary frames, which provide additional control over how to describe a data set linguistically. Some of these are illustrated in Table 4. For example, the entry "cuisine" in the table would first attempt to enumerate all cuisines available (using the "enumerate" template), but only if the total count is below a threshold specified by the developer. If that fails, it then attempts to speak about a majority of restaurants (the "majority" template), again, if an adequate percentage of the total are contained in a small enough set (e.g., 60% are contained in the first 4 choices; specified by the developer). If all else fails, it simply provides a count on the number of options under "cuisine." Notice that in the example the "street" option does not back off to a count, but rather would be omitted altogether should both options fail. The "singleton" and "singleton_out" notation associated with "price_range" illustrates the summarization capability for unique or missing values that apply across the entire set.

6. Dialogue Simulations

Once we are able to summarize constrained data sets, the next question to address is the coherence of a dialogue produced through expected user interactions. After the initial dialogue

```
{c summary
   :count 14
   :categories
   ( {c cuisine
         :ordered_counts ( 4 2 2 2 ...
         :ordered_values ( “american” “brazilian” “indian” ..)}
   {c price_range
         :ordered_counts ( 7 2 2 )
         :ordered_values ( “cheap” “low” “medium” )}
   {c street
         :ordered_counts ( 4 2 1 ...
         :ordered_values ( “hancock street” ..)} }
```

Table 3: Example summary frame resulting from the refinements for symbolic categories for “restaurants in Quincy.”

cuisine	(>enumerate >majority >count)
street	(>enumerate >majority)
price_range	(>singleton >singleton_out)
singleton	“all of them are” :singleton .
singleton_out	“none of them are” :singleton_out .

Table 4: Selected entries from the GENESIS message file to control paraphrasing of summary frames. The developer has control over parameter settings in a separate table.

system is in place, but before releasing it to real users, an effective strategy can be to simply simulate the user's turn in the dialogue. We have adopted a method similar to [3] that enables us to easily examine the basic behavior of the system, under a myriad of scenarios. A simulation server has been implemented as part of the Galaxy system, synthesizing possible user inputs. At each turn, the simulator selects at random an item, based on the explicit information given by the system response. First the simulator randomly selects a possible category to consider (such as cuisine), then it randomly picks a possible value for that attribute (such as seafood), weighted by the frequency counts. Through multiple simulated turns the search is narrowed down until the list is small enough to enumerate for final selection.

Table 5 illustrates a simulated dialogue in the restaurants domain. Using the simulator, hundreds of dialogues are generated within minutes, and developers can iteratively spot and repair system weaknesses. In the current restaurant domain, the simulator arrives at a single choice within on average 4.6 turns. By running more than a thousand dialogues, we have successively improved various aspects of the responses generated, given the multitude of scenarios possible in a large data set.

7. Conclusions and Future Work

This paper has introduced a number of approaches that ease the effort required to create a dialogue system and provide a dialogue management and response planning strategy that is adaptable to the on-line content, thereby improving the interaction with the user. We also introduce the use of a simulation server for examining the operation of the overall dialogue system, particularly the interaction between the dialogue flow and the response generation outputs. In the future, one direction is to expand the simulation capabilities in order to debug and evaluate more complex dialogues involving a broader class of queries, such as queries against the attributes of a specific database entry. Another direction of interest is the use of simulation to rapidly synthesize vast amounts of data that can be used as initial training material for the recognizer language model and natural language component. Since GENESIS can generate different paraphrases for the same user input query (illustrated as “PARA” in Table 5), it is possible to obtain a relatively rich corpus of simulated user utterances this way. This holds the promise for configuring robust language components for the system, beginning with no training data, and given only the database contents and their structure. Our future plans are to evaluate this paradigm.

SIM:	I would like a restaurant.
PARA:	What restaurants do you have?
SYS:	I have found 983 restaurants. Most of them are located in Boston and Cambridge. There are 32 choices for cuisine. I also have information about price range.
SIM:	Boston.
PARA:	Okay tell me about the ones in Boston.
SYS:	I have found 401 restaurants in Boston. There are 29 choices for cuisine.
SIM:	Seafood.
PARA:	Do you have any that serve seafood?
SYS:	I have found 19 seafood restaurants. They are predominantly in Back Bay, the North End, South Boston and the South End.
SIM:	South End.
PARA:	In the South End.
SYS:	I have found 3 restaurants. Grill Fish on Columbus Avenue, McCormick & Schmick's on Columbus Avenue and Joseph's on High on High Street.
SIM:	Grill Fish.
PARA:	The Grill Fish please.

Table 5: Example of a simulated dialogue narrowing down the data set from all restaurants in Boston to the “Grill Fish.” The simulated inputs (SIM) are paraphrased into a number of variants (PARA) by GENESIS and processed to produce system responses (SYS).

8. Acknowledgements

The authors wish to thank Eric Wagner for writing the initial implementation of the clustering code.

9. References

[1] L. Baptist and S. Seneff, “GENESIS-II: A Versatile System for Language Generation in Conversational System Applications,” *Proc. ICSLP*, II, 271–274, Beijing, China, 2000.

[2] J. Glass and E. Weinstein, “SPEECHBUILDER: Facilitating Spoken Dialogue System Development,” in *Proc. Eurospeech*, 1335–1338, Aalborg, Denmark 2001.

[3] R. Lopez-Cozar *et al.*, “A New Method for Testing Dialogue Systems Based on Simulations of Real-World Conditions,” *Proc. ICSLP*, 305–308, Denver, Colorado, 2002.

[4] J. Polifroni and G. Chung, “Promoting Portability in Dialogue Management,” in *Proc. ICSLP*, 2721–2724, Denver, Colorado 2002.

[5] S. Seneff, “TINA: A Natural Language System for Spoken Language Applications,” *Computational Linguistics*, Vol. 18, No. 1, 61–86, 1992.

[6] S. Seneff *et al.*, “Galaxy-II: A Reference Architecture For Conversational System Development,” *Proc. ICSLP*, 931–934, Sydney, Australia, 1998.

[7] A.R. Toth *et al.*, “Towards Every-Citizens Speech Interface: An Application Generator for Speech Interfaces to Databases,” *Proc. ICSLP*, 1497–1500, Denver, Colorado, 2002.

[8] V. Zue and J. Glass,“Conversational Interfaces: Advances and Challenges,” *Proc. IEEE*, 88(8), 1166–1180, 2000.