Scalable and Portable Web-Based Multimodal Dialogue Interaction with Geographical Databases

Alexander Gruenstein, Stephanie Seneff, and Chao Wang

Spoken Language Systems Group
MIT Computer Science and Artificial Intelligence Laboratory
The Stata Center, 32 Vassar Street, Cambridge, MA 02139, USA
{alexgru,seneff,wangc}@csail.mit.edu

10.21437/Interspeech.2006-145

Abstract

We describe work towards developing a scalable and portable framework for enabling map-based multimodal dialogue interaction over the web. Working in the context of a restaurant-guide system, we show how large information databases harvested from the web can be accommodated in our speech recognizer, parser, and web-based GUI. We compare two dynamic language modeling techniques, which calculate context-dependent weights for the large sets of proper nouns associated with geographical entities such as restaurants and streets. We show that the more fine-grained approach results in a 7.8% reduction in concept error rate.

Index Terms: multimodal dialogue system, language modeling, restaurants, maps, world wide web

1. Introduction

In building multimodal dialogue systems for database access, it is important to focus on both the *portability* and the *scalability* of their infrastructure. By portable, we mean that the infrastructure should be independent of the type of entities constituting the underlying database. In practice, this might mean that the same infrastructure could be used to build dialogue systems for a particular "class" of databases – in this paper, we focus on the class which contains geographically situated entities such as restaurants, hotels, and apartments. By scalable, we mean both that the same infrastructure should work equally well for small and large databases, and, in the case of geographical databases, that the infrastructure should support easily switching among different geographical regions – such as, for example, among different major metropolitan areas. Issues of both portability and scalability apply to all of the major technology components of dialogue systems: speech recognition, language understanding, dialogue management, language generation, speech synthesis, and the graphical user interface (GUI). In previous research, we have shown progress with regard to the portability of dialogue management [1], and we have shown how cross-domain fertilization and simulation can create portable language models for speech recognition [2, 3].

In this paper, we show how a restaurant guide dialogue system can be *scaled* to handle a large database of restaurants harvested from web sources. While our lab has developed several iterations of a restaurant-guide dialogue system [4], and other groups have also developed interesting applications backed by geographical databases (*e.g.* MATCH [5], AdApt [6]), the system described in this paper stands out in that it both has comparable coverage to commercially available web-based databases, and it can seamlessly switch among different metropolitan areas. A critical factor in such genericity is the effective contextualization of proper nouns such as restaurant, city, neighborhood, and street names. This depends on the recognizer's ability to rapidly reconfigure itself to support a subset of the full vocabulary that is appropriate given the immediately preceding dialogue interaction. Each metropolitan region – and, in fact, each city and neighborhood within each metropolitan region – is associated with a specific set of licensed proper nouns, which can be swapped in and out of the recognizer dynamically at each dialogue turn. The system can be configured such that the set of dynamically available nouns is based only on the metro-region under discussion, or such that the nouns are swapped with more fine-grained control based on the current dialogue context. Furthermore, because all proper nouns are encapsulated in dynamic classes, language model training data for one metro region are applicable generally to all regions, as the data consist mostly of the "glue" language encompassing the proper nouns. Despite the fact that the individual proper nouns do not appear in the training data, appropriate within-class probability distributions can be derived based on frequency counts in the database. Finally, because the recognizer tags these proper nouns in its output (*e.g.*, *<$city> Menlo Park </$city>*), the parser can be agnostic to the contents of the proper name classes.

While the methods we discuss in the paper are applied to the restaurant-guide domain, we believe they are *portable* to other

U_1:	Show me Greek restaurants in Boston.
S_2:	There are six Greek restaurants in Boston. [*shown on map*]
U_3:	What are the hours for Steves?
S_4:	Here are the hours for Steves Restaurant: From Monday to Saturday 7:30am - 11:00pm, Sunday 10:00 am - 10:000pm.
U_5:	Are there any Italian restaurants along this street. [*draws line*]
S_6:	There are 12 Italian restaurants along this street. [*shown on map*]
U_7:	Show the web page for this one. [*circles a restaurant*]
S_8:	OK. [*displays web page from which the data was harvested*]
U_9:	I'd like to add a landmark here. [*clicks mouse on location*]
S_{10}:	OK. [*dialogue box displayed for user to type name*]
U_{11}:	[*user types: "Fenway Park"*]
S_{12}:	OK. I have added a landmark named Fenway Park. [*shows it*]
U_{13}:	Are there any cheap restaurants near Fenway Park?
S_{14}:	There are 12 inexpensive restaurants near Fenway Park.[*displayed*]
U_{15}:	[*Clicks drop down menu to select "San Francisco"*]
U_{16}:	Are there any cheap restaurants near 100 Santa Cruz Avenue in Menlo Park?
S_{17}:	There are five inexpensive restaurants near there. [*shown on map*]

Figure 1: *An example interaction, labeled with U for user, S for system. Gestures and system actions are bracketed. Some system remarks were shortened for brevity.*

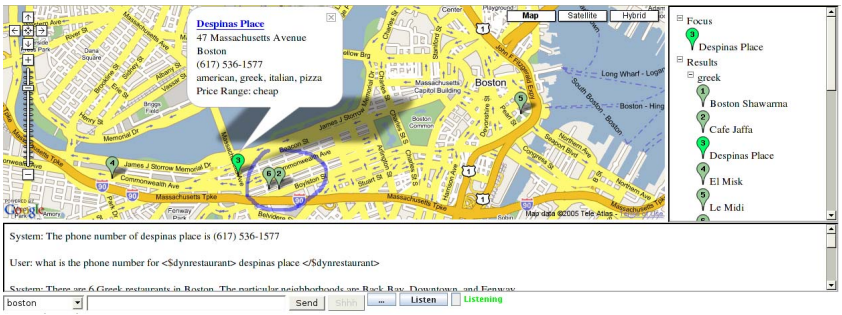

Figure 2: *Screenshot of the restaurant guide user interface.*

multimodal database-access applications as well. Of particular note is that the GUI is quite generic in its ability to provide a multimodal interface to geographically situated entities. It consists of a dynamic web page which is accessible via any web browser, and accepts multimodal input in the form of speech, typed input, and mouse (or pen) gestures such as circles, lines, and clicks on particular items. It displays the system's responses as text and plays back synthesized speech. To the best of our knowledge, this is the first multimodal dialogue system accessible entirely via the web. An example interaction with the system is provided in figure 1, while figure 2 shows a screenshot of the multimodal interface.[1]

The remainder of this paper is organized as follows: section 2 describes the automatic procedure to harvest a database from web sources. Section 3 describes the overall system in more detail, highlighting features that enable the example dialogue in figure 1. Section 4 provides a system evaluation based on data collection experiments, comparing two different dynamic language modeling techniques. Section 5 concludes and looks to the future.

2. Harvesting the Database

The data for the system are gleaned from crawling restaurant databases available on the web. We currently have data for seven major metropolitan areas in the United States: Austin, Boston, Chicago, Los Angeles, San Francisco, Seattle, and Washington D.C. Acquiring more data is a fully automated process, which begins with a custom-built web spider scraping restaurant information from sources on the web. Depending on the metropolitan area, this data may include anywhere from around 50 to 250 nearby cities, containing from 3,000 to 17,000 restaurants.

A sequence of steps then transforms the harvested data into a structured database suitable for generic filtering, as well as into an exhaustive proper noun vocabulary relevant for interacting with that database. An aspect that is often overlooked is the somewhat tedious task of verifying that the harvested proper nouns are well-formed and adequately cover the expected usage model. The speech recognizer and synthesizer both rely on a letter-to-sound system to generate appropriate pronunciations for any uncommon extracted words. Thus an abbreviation like "pzzr" has to be converted to its full form, "pizzeria," to be properly processed. The clean-up phase is governed by a generic set of rules specified in a configuration file. The rules are contextualized to the database field and support special instructions for match condition and appropriate action. Aside from certain special cases (such as "st" in "st lorraine" which should be rewritten as "saint" instead of "street"), a large percentage of the clean-up operations can be applied generically to restaurant data for all cities.

Original string harvested from the web:

```
"lunch tue-fri 11:30am-1:30pm;
dinner mon-thu 5:30pm-9:30pm
fri 5:30pm-10pm sat ... "
```

Structured representation:

```
{c hours
   :meal "lunch" :days "tu we th fr"
   :start_time 1130   :end_time 1330
   :and {c hours  :meal "dinner"
          :days "mo tu we th"
          :start_time 1730 :end_time 2130
          :and REMOVED FOR BREVITY ... }}
```

Figure 3: *Input format of the harvested hours information (top) and resulting structured representation after parsing and processing, abbreviated (bottom).*

The next step is to generalize the restaurant names, since users will often not refer to a restaurant by its entire name as specified in the database. Appropriate aliases need to be algorithmically generated to augment the recognizer's vocabulary. While it is logical that words like "restaurant" and "grill" can be omitted, one has to take care not to generate "bar and" from "bar and grill," or license an alias that is dangerously confusable, for example, licensing "Boston" as an alias for "Boston Cafe." Again, a framework is developed which enables the developer to characterize the appropriate rules succinctly, and exceptions can be noted.

The most sophisticated of the database processing procedures is the one that handles the hours, which need to be fully parsed to obtain an appropriate meaning analysis. Figure 3 illustrates the processing of a restaurant "hours" field into a structured key-value representation. A context free grammar handles most of the different observed variants on abbreviated hours strings, producing a structured format which allows the system to filter on questions like, *Are they open for lunch on Sunday?* Language generation rules paraphrase from the structured form into fluent English for system replies concerning hours information.

3. System Architecture

The overall system architecture is depicted in figure 4. The interface to the system is presented via a client-server model in which the user navigates to a web page that presents the user interface. Speech, typed, and gesture inputs are communicated to the server, where all dialogue processing occurs. The server then sends synthesized speech directly to the client where it is played locally, and it communicates database query results to the client via an intermediary servlet, where they are displayed on the dynamic web

[1] A video demonstration of the system can also be viewed online at this URL: `http://www.mit.edu/~alexgru/rest-videos/`

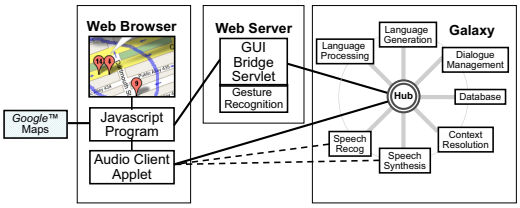

Figure 4: *Architecture Diagram*

page in the user's browser. On the server side, the Galaxy architecture [7] provides for a hub-based interaction among the speech recognizer, synthesizer, parser, context resolution server, dialogue manager, and database.

3.1. GUI and Audio Front-End

A screenshot of the GUI appears in figure 2. It consists of a dynamic web page centered around a *Google*™map, and can be accessed via any web browser. The application interacts with the map programatically via a publicly available application programming interface.[2] Messages are passed between the GUI web page and the dialogue infrastructure using AJAX[3] techniques, allowing a high degree of responsiveness not typically found in web applications. On the web server, a GUI Controller Servlet moderates traffic between the front-end web page and the Galaxy architecture used by the other dialogue system components. Audio communication is controlled via a Java applet embedded in the page which provides a push-to-talk button and endpointing. Speech and synthesized audio are streamed across the network as needed, as the dotted lines in the diagram indicate.

The GUI can display any type of geographically situated entity, and as such serves as a generic map-based front-end. In the restaurant-guide application, such entities include restaurants, cities, and user-defined landmarks. Currently in-focus entities are displayed to the right of the map as either a simple list, or sorted into a tree structure based on attributes.

Finally, the user can draw on the map while speaking, permitting multimodal commands like those in figure 1. Visible entities may be circled, as in U_7. Strokes that form points or lines are recognized, as illustrated in U_5 and U_9. Arbitrary regions may also be circled, allowing commands such as *Show cheap restaurants here*. Gesture recognition is performed using the algorithms described in [8].

3.2. Linguistic Processing

In this section we briefly describe the various system components that are involved with understanding and interpreting the user's questions. The interested reader can consult the provided references for further detail.

Speech Recognition: The SUMMIT system, specifically the version in [4] which supports dynamic language model manipulation, is used. The n-gram language model contains several dynamic classes: *$restaurant*, *$street*, *$city*, *$neighborhood*, and *$landmark*, whose contents may change over the course of dialogue interaction, reflecting dialogue context. Each class is a finite state transducer (FST) which can be sewn into the parent recognizer's main FST at any time, expanding into the appropriate vocabulary whenever that class appears in the upper layer of the FST (see [9]). Each class is automatically tagged in the output string to support linguistic analysis.

Language Understanding and Language Modeling: For language understanding we utilize a lexicalized stochastic syntax-based context free grammar, specialized for database query applications. The recognizer's class n-gram language model is automatically derived from the grammar, using techniques described in [10]. Both static and dynamic classes are specified. While the words populating dynamic classes are constantly changing during a dialogue, the NL grammar remains static by utilizing the tags provided by the recognizer to aid in parsing and assigning proper roles to the (unknown) proper nouns. The n-gram weights have been trained using a small corpus of transcribed developer interactions with the system.

Multimodal Context Resolution: All entities introduced into the discourse either verbally (by the user or by the system) or through gesture are added to a common discourse entity list. A heuristic algorithm described in [11] resolves plausible anaphors for deictic, pronominal, and definite noun phrases.

Response Planning and Generation: A generic dialogue manager [1] filters the database based on user-supplied constraints and prepares a reply frame encoding the system's intended response, which is then converted to a surface-form string using the GENESIS generation system [12] and finally to a synthesized waveform using a commercial speech synthesizer. A separate multimodal reply frame is converted to XML by GENESIS to update the set of entities shown on the GUI.

3.3. Dynamic Language Modeling

As mentioned above, the language model contains *dynamic classes*, whose contents can be manipulated prior to recognizing each utterance based on the current context. The *$landmark* class provides a straightforward example of how the contents of a dynamic class may change. Utterances 9-14 in figure 1 demonstrate how a user may enroll a new landmark into the system's vocabulary by adding it to the *landmark* dynamic class, and refer to it in subsequent interactions.

The system can be run in two different modes with regard to how it populates the other four dynamic classes. In the first mode, which we'll refer to as *per-metro*, the contents of the *$restaurant*, *$street*, *$city*, and *$neighborhood* classes are changed only when the user selects a different metropolitan region: all restaurants, streets, cities, and neighborhoods in the region are loaded into the classes. In the second mode, which we'll call *context-specific*, the contents of the *$restaurant* and *$street* classes are manipulated in a more fine-grained manner: they are updated at each user turn, depending on the current restaurants, cities, neighborhoods, and multimodally defined regions under discussion. For example, following U_5 in figure 1, only the names of the six Greek restaurants displayed on the map will be loaded into the *$restaurant* class. Similarly, only the names of streets in the city of Boston will be loaded into the *$street* class. In addition, if the system detects that the user wants to change the context at hand by, for instance, mentioning a city not currently in focus, then the system will dynamically reconfigure the language model before running a second recognition pass over the same utterance to choose a new best hypothesis; an approach previously shown to be effective in [13]. An example of such a two-pass recognition occurs for U_{16} in figure 1. The result from the first pass "*Are there any cheap restaurants near 100 $street_oov in Menlo Park*," utilizes the out-of-vocabulary (OOV)

[2] See `http://www.google.com/apis/maps`

[3] Asynchronous Javascript and XML

	WER	CER
per-metro	27.1%	30.4%
context-specific	26.1%	28.0%
relative reduction	3.7%	7.8%

Table 1: *Word/Concept error rates*

model for the street name. Having identified *Menlo Park* as a new city, the system loads the streets and restaurants in *Menlo Park*, and performs a second recognition pass.

Finally, counts from the database of the number of restaurants on each street, and in each city or neighborhood are used to weight the contents of these classes. For example, the weight of a particular street name is proportional to one plus the number of restaurants on that street, normalized such that the weights sum to one. Pilot experiments showed that such weighting resulted in slightly lower error rates than did uniform weighting.

4. Evaluation

We performed a preliminary performance evaluation by logging the interactions of 10 subjects who were asked to find at least four appealing restaurants where they hadn't eaten before. Two of the restaurants were intended to be in the greater Boston area, while the other two could be elsewhere. We collected 546 user utterances, which were transcribed. A few users had interacted with the system quite a few times, while the rest had never used the system. Nine of the subjects (90%) completed the tasks.

We analyzed the user data to compare the performance of the two different modes of dynamic language modeling described in section 3.3. *A priori*, it is difficult to know which mode should perform better. While the *per-metro* mode allows the user to say any restaurant or street name at any time, it is natural to expect that the *context-specific* mode would more accurately recognize streets and restaurants in focus as well as reduce the "misfiring" of non-relevant restaurant and street names. The vocabulary excluding all proper nouns is about 1050 words; however, the number of proper nouns can exceed 30,000 in the *per-metro* mode.

The data were collected using the *context-specific* language model configuration, and then the recorded utterances were recognized again in an offline process using the *per-metro* language model configuration. Table 1 shows the overall word error rates (WER) and concept error rates (CER) in each condition. The CER was evaluated on a subset of 457 utterances (83.7% of the test data), whose transcripts could be parsed to provide a reference. There are in total 22 types of concepts, and each parsable utterance contains 2.2 concepts on average.

While the difference between the two systems is not pronounced, the *context-specific* system showed modest relative reductions in word and concept error rates. In analyzing the differences between the two language models, we found that users sometimes did refer to restaurants and streets not currently deemed by the system to be "in focus," and which, hence, were not supported by the *context-specific* language model. This happened often when they were interested in a particular major street, or already knew the name of a restaurant in the metropolitan region. It also occurred if they asked about a restaurant or street which was recently discussed, but no longer deemed to be in focus. We believe further gains could be made by loosening the restrictions on the *context-dependent* language model – *e.g.*, all proper nouns could be active at once; however, the in-focus set could have a much higher share of the probability mass. The weights of recently discussed entities could also decay over time, rather than immediately being set to zero.

5. Conclusion

We have discussed how generic techniques can be used to harvest a database from the web, and then integrate it into a scalable infrastructure for a multimodal dialogue system. We have demonstrated that scalable language processing components can be made agnostic to the particulars of each database entry. Moreover, we have compared two generalizable techniques for dynamic language modeling in applications that access large databases, and have shown how database contents can be used to assign appropriate within-class weights to proper nouns. Finally, our architecture is portable to other geographical database database applications, particularly the map-based multimodal GUI.

6. Acknowledgments

This research is sponsored by the T-Party Project, a joint research program between MIT and Quanta Computer Inc., Taiwan.

7. References

[1] J. Polifroni, G. Chung, and S. Seneff, "Towards the automatic generation of mixed-initiative dialogue systems from web content," in *Proc. of EUROSPEECH*, 2003.

[2] G. Chung, S. Seneff, and C. Wang, "Automatic induction of language model data for a spoken dialogue system," in *Proc. of SIGdial*, 2005.

[3] C. Wang, S. Seneff, and G. Chung, "Language model data filtering via user simulation and dialogue resynthesis," in *Proc. of INTERSPEECH*, 2005.

[4] G. Chung, S. Seneff, C. Wang, and L. Hetherington, "A dynamic vocabulary spoken dialogue interface," in *Proceedings of INTERSPEECH*, 2004, pp. 327–330.

[5] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor, "MATCH: An architecture for multimodal dialogue systems," in *Proc. of ACL*, 2002.

[6] J. Gustafson, L. Bell, J. Beskow, J. Boye, J. E. Rolf Carlson, B. Granström, D. House, and M. Wirén, "AdApt a multimodal conversational dialogue system in an apartment domain"," in *Proc. of ICSLP*, 2000.

[7] S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue, "Galaxy-II: A reference architecture for conversational system development," in *Proc. ICSLP*, 1998.

[8] S. B. Wang, "A multimodal Galaxy-based geographic system," M.S. thesis, MIT Department of Electrical Engineering and Computer Science, 2003.

[9] J. Schalkwyk, I. L. Hetherington, and E. Story, "Speech recognition with dynamic grammars using finite-state transducers," in *Proc. of EUROSPEECH*, 2003.

[10] S. Seneff, C. Wang, and T. J. Hazen, "Automatic induction of n-gram language models from a natural language grammar," in *Proc. of EUROSPEECH*, 2003.

[11] E. Filisko and S. Seneff, "A context resolution server for the Galaxy conversational systems," in *Proc. of EUROSPEECH*, 2003.

[12] L. Baptist and S. Seneff, "Genesis-II: A versatile system for language generation in conversational system applications," in *Proc. of ICSLP*, 2000.

[13] I. L. Hetherington, "A multi-pass, dynamic-vocabulary approach to real-time, large-vocabulary speech recognition," in *Proc. of INTERSPEECH*, 2005.