Releasing a Multimodal Dialogue System into the Wild: User Support Mechanisms

Alexander Gruenstein **Stephanie Seneff**
Spoken Language Systems Group
M.I.T. Computer Science and Artificial Intelligence Laboratory
32 Vassar St, Cambridge, MA 02139 USA
{alexgru,seneff}@csail.mit.edu

Abstract

We present *City Browser*, a web-based platform which provides multimodal access to urban information. We concentrate on aspects of the system that make it compelling for sustained interaction, yet accessible to new users. First, we discuss the architecture's portability, demonstrating how new databases containing *Points of Interest* (*POI*s) may easily be added. We then describe two interface techniques which mitigate the complexity of interacting with these potentially large databases: (1) context-sensitive *utterance suggestions* and (2) *multimodal correction* of speech recognition hypotheses. Finally, we evaluate the platform with data collected from users via the web.

1 Introduction

Multimodal dialogue interfaces, which provide a graphical input and output modality in addition to speech, do not currently tend to be available to the wide audience of users that can be found for more traditional, telephone-based speech-only dialogue systems. At the moment, most development and testing of such systems occurs in the laboratory, under controlled experimental conditions. In this paper, we focus on efforts to convert our restaurant-guide multimodal dialogue system previously described in (Gruenstein et al., 2006; Gruenstein and Seneff, 2006) into *City Browser*, a full-fledged platform for providing urban information multimodally via the world wide web. Because *City Browser* is available via the web, it has millions of potential users on all sorts of Internet-connected devices, which may or may not have keyboards. However, it is a major challenge to actually reach out to these users with an interface that is compelling and capable enough to afford a sustained interaction, yet accessible and intuitive enough to be usable by people who likely have no past experience with multimodal dialogue systems.

In this paper, we identify a core set of capabilities which make *City Browser* compelling as a generic platform for presenting geographic information. The platform provides capabilities to support multimodal exploration of databases containing *Points of Interest*. Exploration is enhanced by allowing users to access information about public transportation, obtain driving directions, and locate addresses on the map. However, over the course of developing the system, it has become apparent that, even as the platform becomes more useful, it also tends to become more difficult to use – a trend often noted by dialogue system designers.

We present two novel user-interface components which are intended to make multimodal dialogue systems more usable in the face of growing complexity. The first is a *suggestions module* which takes advantage of the visual modality to provide high-quality, context-sensitive suggestions to the user about what she can say or do next. The second is a *multimodal error correction framework*, which provides the user with an interactively correctable N-best list of recognizer hypotheses.

Finally, because our interest is in understanding how real users interact with multimodal dialogue

Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue, pages 111–119,
Antwerp, September 2007.

systems outside of the laboratory environment, we describe our nascent, web-based data collection efforts in which users interact with *City Browser* from their own computers. In particular, we focus our analysis on the response of naive users to the presence of the suggestions module and correctable N-best list.

2 A Platform for Accessing Urban Information

The *City Browser* platform grew out of our work with a multimodal dialogue system which was initially restricted to information about restaurants. The system's overall client-server architecture for speech recognition, linguistic processing, and gesture interpretation has previously been described in detail (Gruenstein et al., 2006). The interface is web based and users need only a web browser equipped with the Java plug-in to access the system. The interaction is centered around a map, as pictured in the screenshot in Figure 5 (in the appendix). In addition to speech input, users can draw on the map, as well as click on displayed *Points of Interest* (*POI*s). An example of a dialogue with the system is given in Figure 1. *City Browser* currently has knowledge of 10 major U.S. metropolitan areas, typically encompassing 20-50 miles around a major city. Harvesting new data from the web for additional metropolitan areas is a semi-automatic process [see (Gruenstein and Seneff, 2006)].

As we observed users interact with our prototype containing only restaurant information, we realized that, while perhaps of some interest, the system was not useful enough for a genuinely compelling interaction. Several frequently requested features were unavailable. In particular, users wanted to find an address on the map and then search near it, obtain driving directions between points of interest, and find out about available public transportation options.

City Browser expands on our previous prototypes by providing these new features. Users can now plan an outing by finding a restaurant of interest and then figuring out how to drive or take public transportation to get there; or they can look for an address, find out how to get there, and look for restaurants nearby. Moreover, when using an address to anchor a search,

S_0:	Welcome to City Browser
U_1:	Show me 32 Vassar Street in Cambridge.
S_2:	Here is 32 Vassar Street in Cambridge. [*shown on map*]
U_3:	Are there any Indian restaurants near here?
S_4:	There are 10 Indian restaurants near 32 Vassar Street in Cambridge. [*results displayed*]
U_5:	What is the nearest subway stop to India Pavilion?
S_6:	Central square station is located at Massachusetts Avenue and Western Avenue, Prospect Street, and Magazine Street. It is on the Red Line.
U_7:	Are there any museums in this area? [*circles an area of the map*]
S_8:	The Museum of Fine Arts is located at 465 Huntington Avenue in Boston.
U_9:	How do I get to there from 77 Massachusetts Avenue in Cambridge?
S_{10}:	Here are driving directions from 77 Massachusetts Avenue in Cambridge to the Museum of Fine Arts. [*gives link to directions*]

Figure 1: An example interaction. U_i: User; S_i: System. Gestures and system actions are bracketed. Some system remarks were shortened for brevity.

it need not be fully specified, as in *32 Vassar Street, Cambridge, Massachusetts*. Instead, it might only be a street name (*Vassar Street in Cambridge*), or just a city (*Cambridge*).

In addition to these core map-based functionalities, it was also apparent that users wanted to be able to access POIs besides restaurants: they were especially interested in POIs such as tourist attractions, banks, parking garages, and gas stations. In order to support this, we have moved from providing access to a restaurant database, to creating a more generic platform for accessing multiple types of POI databases at once. Given a small amount of metadata and a new database of POIs, the language processing components of *City Browser* can easily be updated to support the new database. In particular, support is provided for databases with some or all of the following attributes: (1) *Name* The name of the POI (*e.g.* Museum of Fine Arts), to be used for natural language generation. (2) *Aliases* Alternative names for the POI, for the language model. (3) *Address or Position* The address of the POI, or a location expressed as a latitude and longitude. (4) *Phone Number* The POI's phone number. (5) *URL* Link to a webpage with more information about the object. (6) *Description* A brief description of the POI.

Our currently deployed version of *City Browser* uses these generic database capabilities to provide access to a database of museums. The architecture

also accommodates the subway station databases for providing public transportation information, the geographical database of cities, streets, and neighborhoods, as well as the existing restaurant database.

2.1 Comparison to Similar Systems

The most similar system we are aware of is MATCH (Johnston et al., 2002), which provided extensive multimodal capabilities for accessing urban information. There is significant overlap between *City Browser* and MATCH. For instance, both provide multimodal access to restaurant and public transit information. A major feature of the MATCH system which is lacking in *City Browser* is handwriting recognition; we have not concentrated on this modality, as we do not currently assume our users will have access to a pen-based interface. Another similar interface is AdApt (Gustafson et al., 2000), which provides apartment rental information in downtown Stockholm.

To the best of our knowledge, *City Browser* stands out in that it provides support for POI databases containing thousands of entries, extending throughout a metropolitan area; in particular, the restaurant databases are comparable in size to those of commercially available, web-based restaurant databases. Moreover, *City Browser* supports a multitude of metropolitan *areas*, rather than just one or two *cities*. As we have just described, it also supports the arbitrary addition of new databases of POIs. *City Browser* provides links to driving directions and supports the recognition of arbitrary addresses with any street name in the metropolitan area. Finally, as noted, *City Browser* is fully web-based; and beyond a web browser, requires only the standard Java plug-in to operate. It is the combination of these factors which make *City Browser* uniquely accessible to a potentially large audience, even as a prototype.

3 Suggestions Module: *What Can I Say?*

City Browser is designed to be a highly *user*-driven interface. The task is generally exploratory in nature, rather than transactional, as tends to be more typical for dialogue systems. In testing earlier iterations of the system, we observed that users often had trouble formulating queries "out of thin air," given their lack of experience using such a system. However, given the large bounds of the system's capabilities, it is difficult to imagine a system-directed dialogue, as there are many paths of exploration.

Natural interaction with increasingly complex and intelligent systems is a fundamental challenge in dialogue system research. As capabilities increase, systems often become much more difficult to use. Users can't easily distinguish an error in which an in-domain phrase is misrecognized, from one in which an out-of-domain phrase is spoken. We utilize *City Browser*'s multiple modalities to gain leverage in attacking this problem, by designing a suggestions module which visually provides users with contextually-specific suggestions as to what they might say next at the current point in the dialogue.

On the right-hand side of the GUI, as shown in Figure 5 of the appendix, we show a list of suggested utterances labeled *What Can I Say?*. In fact, these suggestions extend beyond simply what a user can *say*, by indicating gestures that can be made to accompany certain utterances. As in any dialogue system, particular utterances and actions may only be relevant at a given point in the dialogue; to address this, we have created a module which dynamically produces a relevant set of suggested utterances at each new turn in the dialogue. This serves two purposes. First, it allows us to offer relevant suggestions given the current state of the dialogue, tailored specifically to the current context. Second, even as the same templates are used, their content words (such as city names, street names, and cuisine types) are continually changed, giving the user a general impression of the range of the system's knowledge. For instance, a user might be surprised to see a city 20 miles away from the center of the metropolitan area mentioned, indicating that the system has knowledge of many surrounding suburbs.

Dynamic suggestions, which are dependent on the current dialogue state, are instantiated from handcrafted templates and filled in using the current metropolitan region's POI databases. Suggestions are also tailored to any POIs of interest currently visible on the map. Finally, appropriate follow-up queries are inferred from the user's previous utterance. Figure 2 gives an overview showing how the list of suggestions is generated. The different categories of suggestions generated include the following:

Previous Utterance:	*Show me cheap Indian restaurants in Cambridge*
Key-Value Semantics:	clause=*request*, topic=*restaurant*, cuisine=*indian*, price_range=*cheap*,city=*cambridge*
Matching DB entry (subset of attributes shown):	`{q restaurant :name "india castle" :phone "(617) 864-8100" :streetnum "928" :street "massachusetts avenue" :city "cambridge" :state "ma" :cuisine ( "indian" ) :recommendation "recommended" :price_range "low" :neighborhood "harvard square" }`
Random DB entry:	`{q restaurant :name "dakshin" :phone "(508) 424-1030" :streetnum "672" :street "waverly street" :city "framingham" :state "ma" :cuisine ( "indian" ) :recommendation "*none*" :price_range "low" }`

TEMPLATE	***REALIZATION***
Global	
I'm looking for $PRICE_RANGE $CUISINE restaurants on $STREET in $CITY.	I'm looking for *cheap Indian* restaurants on *Waverly* street in *Framingham*.
What is the nearest $SUBWAYNAME station to $ADDRESS?	What is the nearest *T* station to *672 Waverly Street* in *Framingham*?
Are there any $CUISINE restaurants here? [outline a region with the mouse]	Are there any *Indian* restaurants here? [ouline a region with the mouse]
Subsetting	
Show me the $ATTRIBUTE ones.	Show me the *recommended* ones
Tell me about these. [Circle a few $ENTITY_TYPEs with the mouse]	Tell me about these. [Circle a few *restaurants* with the mouse]
Anaphoric	
What's the phone number of $NAME?	What's the phone number of *India Castle*?
Give me driving directions to $NAME from $ADDRESS	Give me driving directions to *India Castle* from *672 Waverly Street* in *Framingham*
Are there any subway stops near $NAME	Are there any subway stops near *India Castle*?
Contrastive	
What about in $CONTRAST_CITY?	What about in *Framingham*?

Figure 2: This figure shows inputs to the suggestions module, examples of each type of template used to create suggestions, and the actual suggestions which are *realized* by combining each template and the input shown at the top. The inputs to the module are (1) the previous utterance and its key-value semantic representation, (2) the database entries which matched that query, and (3) other randomly selected database entries. This information is used to fill in values in each type of template on the left, yielding the realizations of those templates on the right.

Globally relevant suggestions These are utterances which always apply, such as map commands (*pan right* and *zoom in*), queries about addresses, driving directions, public transportation, and points of interest. The POI databases in the current metropolitan region are used to fill in the templates, as shown in Figure 2. The database entries are used in such a way as to guarantee that each suggested utterance, if uttered (and correctly recognized), will actually yield one or more results. This is very important, since, as some users get to know the system, they read the suggestions verbatim. This helps them to verify that the system is working, and to become more comfortable using it. Figure 2 shows examples of different types of suggestions which might be rendered from a single database entry.

Subsetting suggestions There are two forms for specifying *subsetting* suggestions. First, *multimodal* ones, such as *Tell me about these [Circle a few restaurants with the mouse]*, allow the user to zero in on a smaller set. Second, suggestions which subset by *attribute* show how POI properties can be used to narrow down the set, as in, *Show me the highly rated ones*. A rank-ordered list of properties for each POI type is used for this type of narrowing down; for restaurants we use the *price_range* and *recommendation* properties. Any of these properties which were not mentioned in the user's previous utterance are used to create novel suggestions.

Anaphoric suggestions A user will often want to get more information about a particular attribute of either a single POI in focus or a focus set. We produce two types of suggestions for these cases. If a

single entity is currently salient, we offer *anaphoric* suggestions relating to an attribute of that entity, such as *Tell me its phone number*. For a set of entities, we offer suggestions about the properties of individual members, such as *Can you tell me the address of the Museum of Fine Arts?* In addition to querying about a particular property, users may also use one of the in-focus entities as a reference point for searching for something else, as in *Are there any subway stops close to the Royal East?*

Contrastive suggestions A nice aspect of using natural language to access this type of information is that it is quite easy and natural to build on a dialogue by retaining some attributes of a search query and replacing others. For example, if a user has just said *Can you show me the subway stations in Cambridge*, it is quite natural to follow up with a query such as *What about in Brookline?*. We again use the key-value representation of the user's previous utterance, but this time we look for keys which *were* explicitly mentioned by the user. We then produce suggestions in which one or more of these keys is changed to a different value (which, as usual, is drawn from actual database items). In addition, we offer multimodal contrastive suggestions, such as *What about near here? [Click on a point on the map]*.

Our suggestions system resembles somewhat the multimodal help system developed for MATCH (Hastie et al., 2002). MATCH relied on the user explicitly asking for help, while we offer newly updated suggestions at every turn unobtrusively along the side of the screen. While both the MATCH system and our suggestions system are sensitive to the dialogue context, we are more aggressive about actively incorporating information from the various databases used in the system. We are also more sensitive to the semantic content of previous queries, allowing our module to offer more targeted *subsetting* suggestions. On the other hand, the MATCH system's capability to actually demonstrate how to draw or write during a multimodal command is quite useful, and we hope to incorporate a similar capability in the future.

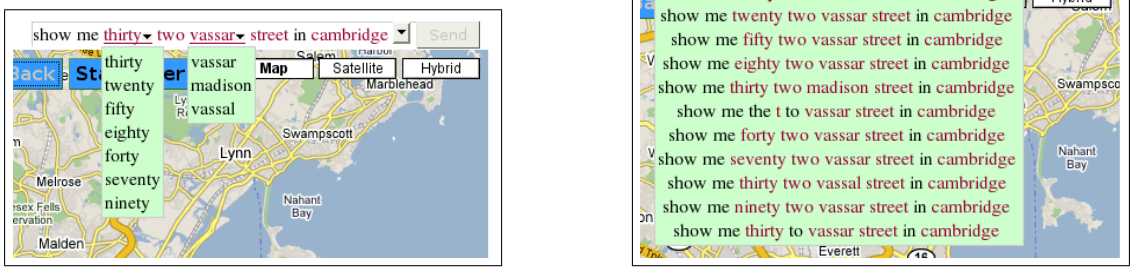
The system can also be seen as providing similar functionality to targeted help systems like those described in (Hockey et al., 2003) and (Gorrell, 2003). However, while these algorithms provide help prompts based on an out-of-domain utterance which was not correctly recognized, the suggestions module described here makes use of the visual modality to try to avoid out-of-domain utterances in the first place. The two approaches could likely be beneficially paired.

4 Multimodal Error Correction

One of the most potentially frustrating aspects of interacting with a dialogue system like *City Browser* is inaccurate speech recognition. Our previous research in this area has focused on dynamic language modeling mechanisms which aim to minimize errors involving proper nouns. Nonetheless, errors arising from the misrecognition of proper nouns are still quite common in *City Browser*, as well as errors having to do with numbers (*e.g.* "thirty" v.s. "fifty"). Other dialogue system designers working in domains with large sets of proper names have also noted this difficulty (Weng et al., 2006).

While extensive research has been performed on multimodal error correction techniques for dictation systems [*e.g.* (Suhm et al., 2001)]– especially with regard to techniques which display alternative hypotheses – we are not aware of dialogue systems which make use of alternatives-based multimodal error correction techniques. Extensive arguments have been made, however, for the potential of multimodal interaction to decrease understanding error rates (Oviatt, 1999).

For *City Browser* we have currently deployed a straightforward mechanism for alternatives-based multimodal error correction, which utilizes the fact that a class n-gram is used as the recognizer's language model – a common mechanism for dialogue system language modeling. Our corrections mechanism presupposes that a large number of errors arise from the misrecognition of content words, rather than the structure of the utterance itself. We display a correctable N-best list which uses semantic knowledge derived from the class n-gram to create alternatives lists. *City Browser* displays the recognizer's top hypothesis, which it has taken to be correct and already responded to, and allows users to correct it in two ways. First, a drop-down menu is available which allows the user to replace the top hypothesis with any of up to 15 of the top hypotheses




```
show me thirty<TENS> two<DIGITS> vassar<STREET> street<STREET_T> in cambridge<CITY>
show me twenty<TENS> two<DIGITS> vassar<STREET> street<STREET_T> in cambridge<CITY>
show me fifty<TENS> two<DIGITS> vassar<STREET> street<STREET_T> in cambridge<CITY>
show me thirty<TENS> two<DIGITS> madison<STREET> street<STREET_T> in cambridge<CITY>
show me the t<SUBWAYNAME> to vassar<STREET> street<STREET_T> in cambridge<CITY>
show me forty<TENS> two<DIGITS> vassar<STREET> street<STREET_T> in cambridge<CITY>
show me seventy<TENS> two<DIGITS> vassar<STREET> street<STREET_T> in cambridge<CITY>
show me thirty<TENS> two<DIGITS> vassar<STREET> street<STREET_T> in cambridge<CITY>
show me twenty<TENS> two<DIGITS> vassal<STREET> street<STREET_T> in cambridge<CITY>
show me ninety<TENS> two<DIGITS> vassar<STREET> street<STREET_T> in cambridge<CITY>
show me thirty<TENS> to vassar<STREET> street<STREET_T> in cambridge<CITY>
...
```

Figure 3: Correctable N-best list. We show a portion of the N-best list generated from the utterance *Show me 32 Vassar Street in Cambridge* along with the drop-down menus available on the user's output. The image on the top-left corner shows what the user sees momentarily during active processing.

which appear on the N-best list. Second, the classes of the language model are leveraged to create potential confusion sets for the members of each class. In particular, whenever a recognition hypothesis is generated by the recognizer, any word or word sequence in the hypothesis which was chosen from one of the language model classes is tagged as such. A separate list is constructed from all words that appear in each class in the top 50 hypotheses on the N-best list. If a class member appears in the top hypothesis, a drop-down menu allows the user to change the value of this class member to that of any other, and then resubmit the altered hypothesis to *City Browser* for processing. Figure 3 shows an N-best list generated by the recognizer, and the resulting drop-down menus which are then available to correct this recognition result.

Typically we expect that this capability would be used primarily to make a single *token replacement* in which one misrecognized class member is replaced with another. We expect that, with less frequency, users will examine the N-best list itself to choose a new *candidate hypothesis*, as this is a more cognitively demanding task. By combining these methods, more complex corrections are possible: a user may first choose a candidate hypothesis with the correct syntactic form, but incorrect class members. They can then perform token replacements to change these class members. This is potentially easier than examining a deep N-best list, as the top-left screenshot in Figure 3 shows. Currently, users can only modify the recognition hypothesis using the provided drop-down menus; though in future work we hope to develop mechanisms which allow the user to type and/or speak to correct parts of the initial hypothesis. However, users are currently free to ignore the correction mechanism by speaking a new utterance.

In our in-lab pilot testing, we realized that users often did not realize that this corrections capability existed, despite help tooltips which point it out. To better advertise the capability, *City Browser* briefly displays each of the available *token replacements* for 1.2 seconds as soon as the recognition hypotheses are available. The benefit here is two-fold. First, it allows the user to easily see if the correct alternative

exists, without having to activate the drop-down list with their mouse. Second, it provides feedback that the system is working even as the input is still being processed and the GUI updated. This both increases the *perceived* responsiveness of the system, and puts the user in a position to detect the error and make the correction more quickly.

5 Preliminary Data Collection Results

We have previously evaluated earlier iterations of the system on several small sets of users using a tablet computer in the laboratory (Gruenstein et al., 2006; Gruenstein and Seneff, 2006). After developing new capabilities, we are now collecting data from users via the web, using their own hardware. We hope that this methodology will enable us to collect a large corpus of data from a wide variety of users, and will allow us to identify issues involved in deploying live dialogue systems.

Subjects are currently being recruited via email lists with an incentive of a $20 Amazon.com gift certificate. Subjects are led through one warm-up task to ensure that their audio set-up is functional, then through 10 scenario-based tasks of generally increasing complexity. The tasks are worded in such a way as to make it difficult to simply "read back" the task description to the system. Several of the tasks are designed to be potentially frustrating if users simply read them back, mentioning concepts that the system does not understand (*e.g.* "highway 93"). This allows us to gather data about how users react when the system encounters out-of-vocabulary words, or concepts the system can't parse or understand. In some cases, it also allows us to collect data about how users might want to interact with the system, if capabilities involving these concepts were available. Figure 6 (in the appendix) shows one of the scenarios used to collect data.

We have transcribed and begun to annotate the data collected from the first 25 users who interacted with the system, and *attempted all*, or *almost all*, scenarios. A total of 1,277 recorded utterances led to recognition hypotheses from these users. The word error rate across all users was 26.0%, similar both to our previous results, and those obtained for small sets of users interacting with MATCH (Bangalore and Johnston, 2004; Johnston et al., 2002) and AdApt (Hjalmarsson, 2002) systems. In order to coarsely gauge the system's performance, we have manually labeled each utterance according to whether the system's response was *entirely correct*, *partially correct* (*e.g.* contained a subset of the information requested), or *incorrect*.

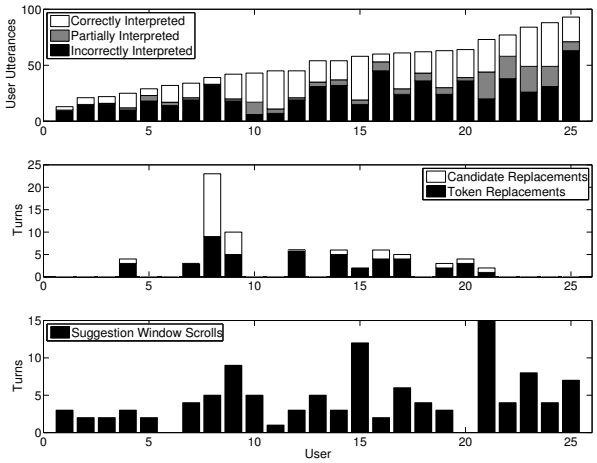


Figure 4: Per-user interaction analysis. Top: correctly, partially correctly, and incorrectly interpreted utterances. Middle: turns with token or candidate corrections. Bottom: turns where suggestions window was scrolled.

Figure 4 shows the number of utterances per user, broken down by the appropriateness of the system's response. Quality of interaction varied quite a bit among users, with some having much more successful interactions than others. We observe that system performance is far from perfect, and are currently further analyzing the causes of the errors. A preliminary analysis shows that audio problems such as inappropriate microphone input level and endpointing errors are responsible for a significant portion of the errors. These types of errors are to be expected when reaching out to a wide range of users using their own hardware, as many users have limited experience using their computer microphone.

We also used log data to glean some knowledge of users' awareness of the N-best corrections capabilities and the suggestions interface. Figure 4 also shows how many times each user used the corrections framework. This is broken down into *token* replacements, in which an individual token (such as a city or street name) was replaced and *candidate* replacements, in which an entirely different candidate

hypothesis was chosen. We found that about half the users (12 of 25) used the corrections capability at least once. In fact, all of these 12 used it more than once.

Finally, to get a very rough idea of whether or not users were at least noticing the suggestions offered by the system, we counted turns in which a user scrolled the suggestions window. The suggestions window can usually fit more than 10 suggestions – depending on screen resolution – when the system first starts. As results are returned, it shrinks to accommodate showing the list of these results, and only the top 5 or so suggestions are usually shown. Users can scroll the window to see all of the currently available suggestions, and this action is logged by the system. Almost all (23 of 25) users scrolled this window at least once; most of them scrolled it during at least several turns. Figure 4 graphs this data. We are encouraged that users are interested enough to scroll the suggestions window, and note that they are likely looking at these suggestions more often than indicated by scrolling, as the top few suggestions (which can be seen without scrolling) are usually intended to be the most relevant to the current context.

6 Summary and Future Work

We have presented *City Browser*, a web-based platform for developing multimodal interfaces which give users access to POI databases. We have shown how *City Browser* can easily accommodate new POI databases. In addition, we have described two aspects of the system which make it easier for users to interact with the unfamiliar technology: a suggestions module and a multimodal error correction interface technique. Finally, we present a preliminary evaluation of these features using data collected from users via the web, using their own computer hardware. We show that users generally do discover and make use of the suggestions feature, while about half use the correctable N-best list.

In the future, we plan to expand the capabilities of *City Browser* based on observations of user interactions and their feedback. We are particularly interested in improving both the suggestions generating and multimodal error correction modules. For example, we believe that a full-blown semantic representation of utterances could be incorporated to allow users to correct structured representations of *City Browser* interpretations rather than text strings.

Acknowledgements

Thanks goes to Chao Wang for input on a draft, Sean Liu for transcription and GUI work, and Liz Murnane for GUI work. This research is sponsored by the T-Party Project, a joint research program between MIT and Quanta Computer Inc., Taiwan.

References

S. Bangalore and M. Johnston. 2004. Robust multimodal understanding. In *Proc. of ICASSP*.

G. Gorrell. 2003. Recognition error handling in spoken dialogue systems. In *Proc. of 2nd International Conference on Mobile and Ubiquitous Multimedia*.

A. Gruenstein and S. Seneff. 2006. Context-sensitive language modeling for large sets of proper nouns in multimodal dialogue systems. In *Proc. of IEEE/ACL 2006 Workshop on Spoken Language Technology*.

A. Gruenstein, S. Seneff, and C. Wang. 2006. Scalable and portable web-based multimodal dialogue interaction with geographical databases. In *Proc. of INTERSPEECH*.

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

H. Hastie, M. Johnston, and P. Ehlen. 2002. Context-sensitive Help for Multimodal Dialogue. In *Proc. of ICMI*, pages 93–98.

A. Hjalmarsson. 2002. Evaluating AdApt, a multi-modal conversational dialogue system using PARADISE. Master's thesis, KTH, Stockhom, Sweden.

B. A. Hockey, O. Lemon, E. Campana, L. Hiatt, G. Aist, J. Hieronymus, A. Gruenstein, and J. Dowding. 2003. Targeted help for spoken dialogue systems: Intelligent feedback improves naive user's performance. In *Proc. EACL*.

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

S. Oviatt. 1999. Mutual disambiguation of recognition errors in a multimodel architecture. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 576–583.

B. Suhm, B. Myers, and A. Waibel. 2001. Multimodal error correction for speech user interfaces. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 8(1):60–98.

F. Weng et al. 2006. CHAT: A conversational helper for automotive tasks. In *Proc. of INTERSPEECH*.

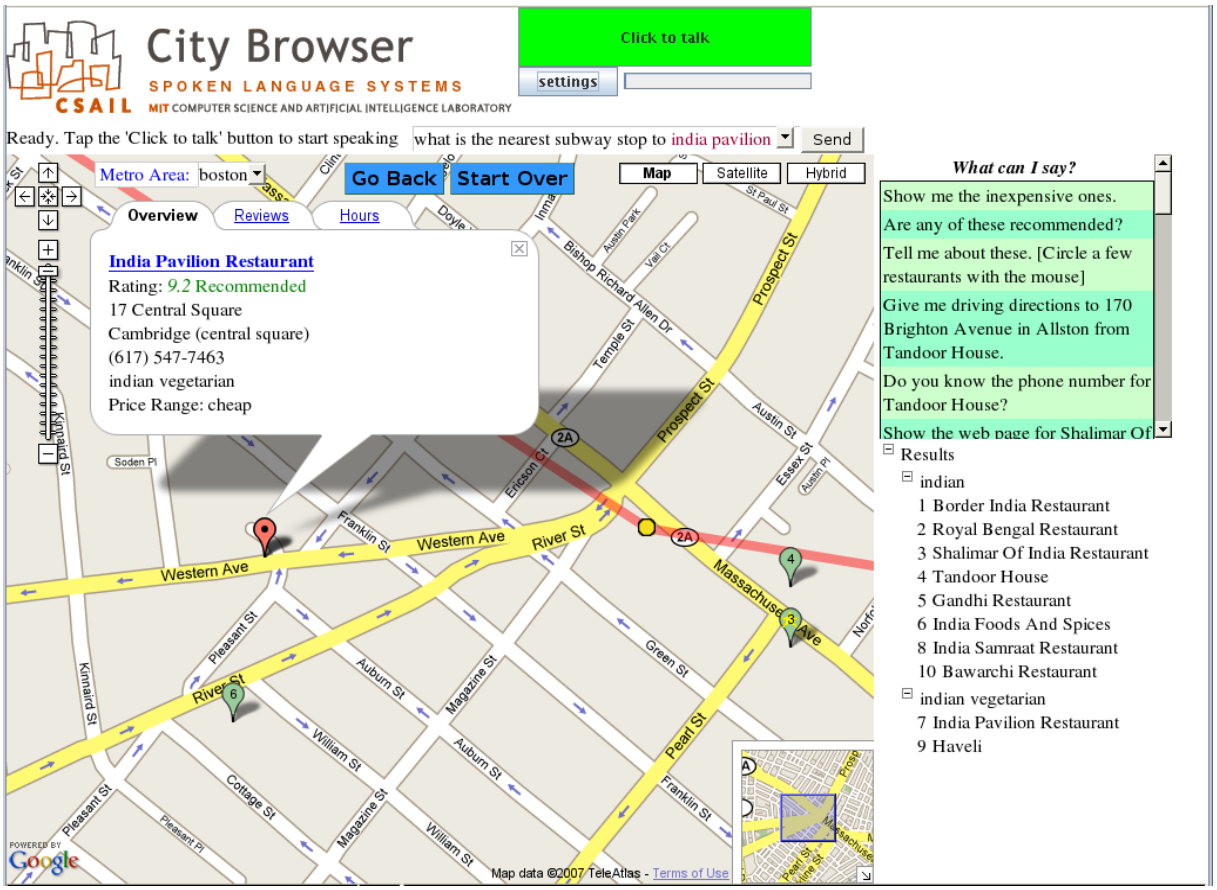


Figure 5: Screenshot of the *City Browser* interface running inside a web browser. At the top, there is a large button that the user presses to start speaking, with a bar underneath which moves as users speak. Immediately below the bar is the top recognition hypothesis for the user's previous utterance, shown as a correctable N-best list. In the upper right corner are the current suggestions of what to say next; below that is a list of restaurants recently returned in an earlier query. These restaurants are shown as the numbered markers on the map at the center. There is also a portion of the overlayed subway map, shown as the line passing through the shaded circle, which has been displayed in response to the user's current query. The shaded circle on that line marks the nearest subway station to the restaurant under discussion, and can be clicked for more information. In the top left corner of the map is a control which allows the user to change the current metropolitan area. To the right of it, are buttons which allow the user to *go back* (undo the previous utterance) and *start over*. The standard Google Maps controls are also overlayed on the map for zooming, panning, and switching to satellite or hybrid view.

You have a friend visiting who wants to go to a couple of different museums in Boston while she's here. She's a sports nut, so you plan to take her to the Sports Museum near the Fleet Center in the morning. Then, you'd like to take her to the Museum of Fine Arts in the afternoon. You are planning on taking the subway to get around starting in Kendall Square. Figure out a plan for doing this. Also, you'd like to find a nice place to eat lunch within walking distance of the Sports Museum, and an Italian place for dinner that is not too far from the Museum of Fine Arts.

Figure 6: Example data-collection scenario