Commute UX: Telephone Dialog System for Location-based Services

Ivan Tashev, Michael L. Seltzer, Yun-Cheng Ju, Dong Yu, Alex Acero
Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
{ivantash, mseltzer, yuncj, dongyu, alexac}@microsoft.com

Abstract

In this paper, we describe a telephone dialog system for location-based services. In such systems, the effectiveness with which both the user can input location information to the system and the system delivers location information to the user is critical. We describe strategies for both of these issues in the context of a dialog system for real-time information about traffic, gas prices, and weather. The strategies employed by our system were evaluated through user studies and a system employing the best strategies was deployed. The system is evaluated through an analysis of 700 calls over a two month period.

1 Introduction

The availability of online maps and mapping software has led to a dramatic increase in location-based services, such as route planning, navigation, and locating nearby businesses, e.g. (Gruenstein, et al., 2006). While much of the effort has been focused on bringing these applications and services to desktop computer users, there is a demand for these services to be available to mobile users.

A significant portion of the mobile users will utilize these services from a vehicle while driving. The automotive environment is a particularly challenging, because operating a vehicle is a hands-busy and eyes-busy task, making the use of touch screens or pointing devices potentially unsafe. In contrast, using of speech as both an input and output modality is a natural and safe means of interacting with information (Tsimhoni, et al., 2004).

In this paper, we present a telephone dialog system called Commute UX (Commute User Experience) that provides location-based information to in-car commuters. The system provides information about traffic, gas prices, and weather, based on real-time data obtained via web services. It was built as a telephone dialog system to enable any user with an ordinary cell phone to access these services without the need for a data plan or additional hardware or software installed in the car. An example dialog is shown in Figure 1.

The most critical part of a dialog system for location-based services is of course, how well the system understands locations spoken by the user. In the proposed system, the definition of location is varied by a context manager, based on the granularity required for a particular task. For example, some tasks may require only knowing the user's current city or neighborhood while others require the system to know the user's precise location. The system engages with the user to obtain the required location information in the most efficient way. In addition, the system can be personalized to a particular user, so common locations or "personal points of interest" are known by the system.

We have found that reliably *conveying* spoken locations to a user is equally important in dialog systems that provide location-sensitive information. We present a method for automatically converting

```
S: Welcome to Commute UX. Would you like
   traffic, gas prices, or weather?
U: I need the closest gas station in Redmond.
S: In that case, I need you to tell me the
   intersection where you are.
U: I'm on the corner of 148th and 35th.
S: You are at the corner of 148th Avenue NE
   and NE 35th Street, right?
U: Yes.
S: The nearest gas station is at the corner
   of NE 24th Street and 148th Avenue NE.
S: What else can I help you with?
U: How's the traffic to Seattle?
S: I'll get the traffic between Redmond and
   Seattle, right?
U: Yes.
S: The traffic between Redmond and Seattle,
   via SR-520, is …
S: What else can I help you with?
U: Nothing, thanks.
S: Good-bye.
```

Figure 1. An example dialog from Commute UX

addresses into more descriptive locations using intersections and landmarks, in a manner that more closely resembles the way in which humans convey location to each other.

The Commute UX system was deployed to a limited group of real users and we evaluated the performance of the system based on an analysis of approximately 700 calls made to the system over an eight-week period. In addition, users of the system were surveyed in order to obtain a subjective evaluation.

2 System architecture

The architecture of Commute UX is depicted in Figure 2. In each turn, the system receives a voice input from the user, processes the input, and reacts to the user accordingly. Six functional modules are involved in this process: the speech recognizer, the semantic parser, the dialog manager, the context manager, the information retriever, and the response manager.

2.1 Speech recognizer

The task of the speech recognizer is to convert the voice input into text, from which semantic information will be extracted and processed. Its performance directly affects the task completion rate and the user satisfaction. Note that the acoustic model used by the speech recognizer is usually independent of the task. However the language model (LM) is highly task-dependent and its quality usually determines the recognition accuracy of the speech recognizer.

The design of the LM is both a science and an art, where a balance needs to be made between the accuracy of the keyword recognition and the flexibility of the speaking style it can support. In our system, we have used a strategy that trains a statistical LM from the slots (e.g., city name, road name, gas type) and information bearing phrases learned from sample queries (e.g. "… the closest gas station in <City> …") and augments it with a filler word N-gram (Yu, et al., 2006) to model the insignificant words. The filler part of the LM absorbs hesitations, by-talk, and other non-information bearing words unseen in the training sentences. The filler word N-gram is pruned from a generic dictation LM.

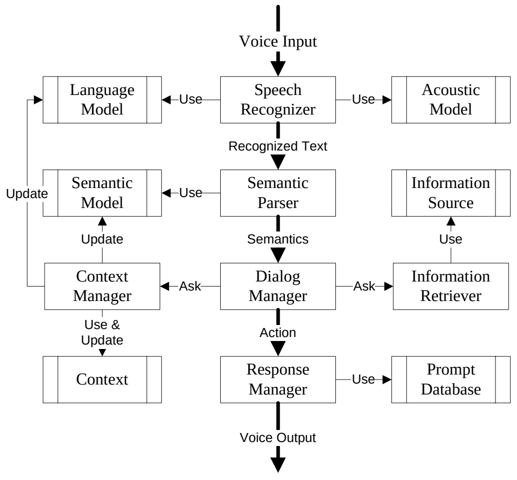

Figure 2. System architecture of Commute UX

2.2 Semantic parser

Semantic parser extracts the semantic information from the recognized text output from the speech recognizer. Converting information into its semantic representation has two benefits. First, semantic representation is more concise and consistent than the phrases. Using semantic representation greatly simplifies the subsequent processing in the later stages. Second, semantic representation is modality independent. By converting information into the same semantic representation, we make the rest of the system isolated from different input modalities. Adding new modalities thus becomes simple and cheap.

Extracting semantic information, however, is not trivial, especially since the output from the speech recognizer contains errors and users may convey multiple semantics in one utterance. The semantic information extracted includes the task classification, which is a generic call-routing problem, e.g. (Kuo, et al., 2002; Carpenter, et al., 1998), and task-specific semantic slots (e.g. origin city, destination city, time of day for weather forecast). Slot labeling is performed using a Maximum Entropy classifier (Berger et al., 1996) trained from the same LM training sentences.

2.3 Dialog manager

The task of the dialog manager is to determine the appropriate actions to take, given the current dialog context and the newly extracted semantic information. Note that both the speech recognizer and the semantic parser are not certain about their results.

The confidence from them needs to be taken into consideration when decision is to be made.

The dialog management is based on a two-level state machine in our system: the turn level and the dialog level. The turn level state machines are pre-built configurable and reusable dialog components such as system-led dialog component and mixed initiative dialog component. These state machines define the basic behaviors of a turn. For example, what to do when the confidence is low, medium, and high, and what to do when silence or mumble is detected. The dialog level (inter-turn) state machine defines the flow and strategy of the top level dialog. For example, what to do if the system cannot get what the user has said after trying twice. In our system, the top level state machine is designed so that it supports both free-form mixed initiative and strict system-led dialog. If the system cannot decipher some of the semantic slots in users' free-form utterances, the system will fall-back to the system-led dialog and guide the user step by step to achieve the user's goal. The user can also yield to the system-led dialog from the very beginning.

The dialog manager gets context information from the context manager and the information requested by the user through the information retriever. The information and prompts are delivered to the user through the response manager.

2.4 Context manager

The context manager plays a key role in Commute UX. Contexts in our system include the user information (e.g., user registered places, user's name, and past requests), the dialog history, and the semantic information confirmed so far. By maintaining current and accurate context information, the context manager can resolve semantic conflicts and make the system synchronous to the user's perceived state.

One important task of the context manager is to update the LM and the semantic model based on the context. By choosing the context dependent LM and the semantic model, the system can greatly reduce the perplexity and achieve higher recognition accuracy and lower number of turns.

2.5 Information Retriever

The information retriever provides an interface between the dialog manager and the backend information sources. In our system, the information is from three major sources: the relatively stable geographical database, which contains information such as cities, streets, intersections, and points of interest (POI); the rapidly changing real time information such as gas prices, traffic conditions, and weather conditions; and the user's registered information such as telephone numbers and personal points of interest (see Section 3.2).

2.6 Response Manager

The response manager presents information to the user or prompts the user for additional information. In our current system, the only presentation modality is voice and so the task of the response manager is to utilize the prompt database, synthesize the best audio output, and present the audio to the user. The system employs several strategies to decide the best manner in which to speak information to the user, as will be discussed in Section 4.

3 Understanding locations from the user

The crux of any dialog system focused on location-based services, such as Commute UX, is to reliably understand the locations spoken by the user. However, the notion of location and the required granularity of location can vary significantly based on the task. For example, for traffic or weather applications, a broad definition of location, such as neighborhood, city, or zip code, can be adequate, e.g. "How's the traffic between Seattle and Bellevue". However, for other tasks such as finding the nearest gas station, or route planning, the user needs to convey a precise location to the system. Finally, there is another distinction between personal locations that can vary based on the user, e.g. home and work, and geographic entities that have standard names and meanings.

3.1 Recognizing: from regions to points

In order to perform recognition of locations, a geographic database is crawled and the relevant information, such as the entity name, entity type and geolocation (latitude/longitude) or bounding box, is stored in a relational database. The database structure enables us to hierarchically categorize locations in a given state: zip codes contain cities, cities contain neighborhoods and points of interest, etc. All of these entities are valid locations in the application and are thus added to the grammar.

When the user makes a query, the parser processes the recognized text and isolates any locations in the spoken utterance. These locations are then passed to the back-end database to find the location data for that entity. The database is searched from most specific location (personal point of interest) to the most general (city or zip code) in order to determine the user's intended location.

In some cases, the task itself dictates the scope of the location grammar. For example, traffic information is only available on major highways, and not local roads. Because we cannot provide a user with traffic information on local roads, a traffic query does not require the same precision in origin and destination as a task such as route planning. As a result, we simplify the task and allow users to make traffic queries only on the roads themselves ("How's the traffic on I-5 north?"), or between cities, neighborhoods, or personal points of interest ("How's the traffic between Bellevue and Seattle?"). This enables the dialog to be much more concise (the user does not have to convey two exact addresses) and because the grammar is more constrained, the accuracy is higher.

There are cases where the user's query can lead to ambiguities. For example, suppose the user asks for the traffic between two cities, and there are two common routes between the origin and destination. Our system will choose the most common route, and attempt to resolve the ambiguity by informing the user of the route it has chosen:

```
U: How's the traffic between Bellevue and
   Seattle?
S: The traffic between Bellevue and Seat-
   tle, via I-90 is light, with an aver-
   age speed of …
```

In this case, the system informed the user that traffic information provided was for the route taking Interstate 90. The user, who presumably knows both routes, can then query for the other route, by asking, "How about via 520?" The context manager maintains the origin and destination cities from the previous query and adds Highway 520 as a road to be included in the route between Bellevue and Seattle. The routing engine will then determine the route between these two cities that takes this highway, and then the corresponding traffic information can be retrieved and delivered to the user.

There are many instances where the user needs to convey an exact location to the system, not simply a city or neighborhood region. For example, if the user needs to find the closest gas station, or would like directions between two places. The most obvious way to convey an exact location is using an address. However, users often do not know a valid address for their current location, especially while they are driving. Even if an address were known, recognition errors make the use of addresses inefficient in conveying location. This was confirmed in (Venkataraman et al., 2003), where an iterative multi-pass approach using a class-based language model was proposed to improve the recognition of spoken addresses. The difficulty is even more apparent when one considers that state-of-the-art recognition accuracy for a five digit number in noise conditions that are realistic for mobile scenarios is about 90%. This means that one out of ten house numbers or zip codes will be misrecognized.

In (Seltzer et al., 2007), we proposed the use of intersections as a convenient and reliable means of conveying location. While the use of intersections alleviates some of problems found in address recognition, it is still a challenging problem. For example, there are over 3500 unique street names, and over 20,000 intersections in the city of Seattle. In addition, streets and intersections are highly acoustically confusable and often spoken informally, with incomplete specifications. For example a user might say "the corner of Third and Denny" rather than "the corner of Northeast Third Avenue and Denny Way".

To reliably recognize intersections, we employ an information retrieval approach. We construct a database of streets and intersections in a particular city. The intersections are treated as documents in a database, and phonetic-level features are derived from the word stings comprising these "documents". When the user utters an intersection, the recognized text is parsed into two street names and the phonetic level features are extracted each street name. Intersection classification is then performed using a vector space model with TF-IDF features. This approach allows the system to reliably recognize intersections in the presence of recognition errors and incomplete street names. Details about this method and an evaluation of its performance can be found in (Seltzer et al., 2007).

3.2 Personal Points of Interest as Locations

One key feature of the Commute UX system is an optional website registration for users. Users can create an account where they provide their phone number and specify any number of personal points of interest (PPOI). These PPOI are specified by a friendly name (e.g. “Jane’s school”), an optional formal name (e.g. “Washington Middle School”), and an address. A back-end web service converts this address to a geolocation and this information is stored in the database. By default, the user is prompted to register home and work as personal locations. Users can then add additional PPOI. Each time a user changes some PPOI, the database is updated and the recognition grammars are regenerated to reflect the current list of unique PPOI friendly names and formal names. When a user calls the system, caller ID is performed as grammar entries corresponding to that user’s PPOI are activated. The caller’s phone number and the recognized PPOI are then used to retrieve the corresponding location form the database.

After a limited internal deployment, we have 276 registered users who created a total of 625 PPOI, but only 97 unique PPOI friendly names in the grammar. The three most popular PPOI were “home”, “work”, and “gym”.

The presence of PPOI also enables the system to assume some default behaviors. For example, if a registered user calls the system during common commuting times, the system will automatically fill the semantic slots with the home and work locations of that user and asks if the user would like the traffic information from home to work (or vice versa).

4 Rendering spoken locations to the user

The ability for the user to understand and remember the locations spoken by the system is as important as the system’s ability to understand the locations input by the user. Conveying locations to users in spoken dialog systems is problematic for several reasons. First, depending on the quality of the TTS voice, understanding a spoken location can be quite difficult, even in optimal conditions. In a vehicle, the environmental noise can make intelligibility even harder. The situation is exacerbated by the high cognitive load required by driving, so the user cannot fully focus on the system’s output speech. In addition, because the user’s hands and eyes are typically busy, s/he cannot write down the location as the system speaks it, and therefore must try to remember the location as closely as possible.

4.1 Automatically rendering locations using intersections and landmarks

To enable users to more easily understand locations, spoken by the system, we modeled the system’s output on the manner in which humans convey locations to each other. For example, a user calling a business to ask its location will often be told by the clerk, “We’re on the corner of 40th and 148th,” rather than “We’re located at 14803 40th Street.” Similarly, humans will often use landmarks, such as “We’re on Main Street near the Shell Station” or “We’re on the corner of Fifth and Mercer, near the Space Needle.

To create a similar capability in our system, we crawled a geographic database containing all streets and intersections along with their latitude/longitude coordinates in a particular city. In addition, we also crawled a database of points of interest (POI), also labeled with their geographic coordinates. These points of interest included a variety of entities, such as schools, libraries, parks, and government buildings. The information about streets, intersections and POI was stored in a database.

Using this information, locations that we want to convey to users, for example the location of a gas stations, are processed as follows. The address of the entity is converted to geographic coordinates. Using these coordinates, the intersections database is queried to find all intersections within 0.05 miles (approximately half a block). If multiple intersections are returned, they are ranked according to an intersection importance metric, defined as the sum of the total number of other intersections of which each constituent street in the given intersection is a member. The top ranked intersection is selected. Following the intersection search, the POI database is queried to identify any POI within 0.1 miles (one block) from the entity of interest.

After this process, each location we can return to the user is represented by its original address, as well as the nearest intersection and/or landmark, if either was found. For those locations that do have a nearby intersection and landmark, we have various ways to present the location to the user summarized in Table 1.

4.2 User preferences for spoken locations

We performed a user study to determine which of these four methods of rendering an address was

Address only	14803 Northeast 51st Street
Address & POI	251 Rainier Avenue North, near Renton Chamber of Commerce
Intersection only	The corner of East Madison Street and 17th Avenue
Intersection & POI	The corner of NE Woodinville Road and 131st Avenue, near City Hall

Table 1. Address representations in Commute UX

Question type	Number	Sum	Accuracy (%)
Address only	67	57.5	85.82
Address & POI	65	53.5	82.31
Intersection only	65	54.0	83.08
Intersection & POI	69	47.7	69.13

Table 2. Recognition rate for various address representations.

preferred by users of a spoken dialog system. Users of the study ran a program on their desktop PCs. Each trial of the study was as follows. The system randomly selected an address from our database of gas station locations. This location was rendered in one of the four styles described in the previous section. The user listened to a TTS engine speak the location. Once the location was spoken, the user was asked to type in as much of location as they could remember. The user could not start typing until the TTS output was complete. The system then randomly chose another address from the database, and rendered it in a style randomly selected from other the three remaining methods. The user again listened to the TTS engine speak the location and had to type in as much of the location as they could remember. After the user completed these two locations spoken in different ways, s/he was asked which, if any, of the two styles was preferred. This completed a single trial of the study. Each user performed a minimum of three trials.

Preferences for location rendering were evaluated based on 40 users who completed a total of 133 trials. The users' data was hand-scored and analyzed in terms of accuracy and user preference. Users' ability to accurately remember spoken locations in these different styles was scored as follows. Addresses and intersections both contain two critical elements (the number and the street name in the former, the two street names in the latter). For locations spoken as addresses or intersections, each element the user correctly identified (within a tolerance of 0.1 miles) is given 0.5 points. Correct recognition of both elements therefore received 1 point. Correct recognition of a spoken POI received 1 point regardless of whether the other elements are correct. Thus, each address transcribed by the user was scored from zero to one in the following way:

$$r = \max\left(\frac{r_1}{2} + \frac{r_2}{2}, r_{POI}\right) \qquad (1)$$

where r_1, r_2 and r_{POI} are either 0 or 1 and are the recognition of the first element, second element, and POI.

The averaged recognition results for each one of the four address representations are shown in Table 2. While the first three representations have approximately the same recognition rate, it is substantially lower for "Intersection & POI". This representation was typically the longest and is therefore the most difficult to remember.

The user preferences are evaluated as follow. For each trial, the preferred representation receives one point. If the user had no preference between the two styles, both are assigned 0.5 points. The final score is weighted with the recognition rate – we weight more these preferences which are properly recognized:

$$p_k = \frac{\sum_i p_i^{(k)} r_i^{(k)}}{\sum_i r_i^{(k)}} \quad k \in [1,4] \qquad (2)$$

where $p_i^{(k)}$ is the preference score of the *i*-th session, where the address is represented in *k*-th way; $r_i^{(k)}$ is the recognition result for the same session, computed by equation (1). Both the non-weighted and weighted average preference scores are shown in Figure 3. Rendering a spoken location using the intersection is clearly preferred, followed closely by the combination of intersection and POI. Because the combination of intersection & POI resulted in the lowest recognition accuracy, we set the system to refer to locations using the nearest intersection whenever possible. In feedback solicited from the users after this study, several participants stated that POI helped only when they were familiar with the area. Otherwise, it was not helpful and added confusion. This indicates that location-based services targeting commuters and residents may want to use POI in describing locations

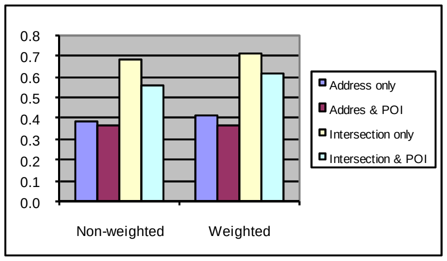

Figure 3. User preferences for address conveying.

to users, while those targeting tourists or business travelers should not.

5 Initial Deployment and Evaluation

The initial version of the Commute UX dialog system can process requests for information about traffic, cheapest and nearest gas stations, and weather in Washington State. The system was demonstrated to approximately 800 Microsoft employees in Redmond, WA campus at the beginning of March, 2007. It was made available to all Microsoft employees but no additional effort was made to actively recruit users. The results presented in this paper are based on an eight week period between March 12, 2007 and May 6, 2007. During this time, a total of 276 users enrolled at the Commute UX website, specifying a phone number and PPOI.

5.1 Analysis of calls

The system received 698 calls during this time period, or 12.5 calls per day. Of these calls, 62.2% were from registered users, while 37.8% were from non-registered users. There were calls from 214 unique phone numbers, of which 55% were registered users. This translates to approximately 3.3 calls per user. However, the distribution of calls per user is not uniform, a 40 users accounted for 50% of the calls during this time period.

From these calls, there were total of 927 tasks that users tried to perform. A task is defined as the user's attempt to obtain a piece of information from the system. In our system, the possible tasks are obtaining a traffic report, the location of the cheapest or nearest gas station, or a weather report. The traffic is the most frequently called with 55% of all queries, followed by the gas prices with 27%, and weather with 17%.

Table 3 shows the average number of turns for each of the three tasks and across all tasks. The results are shown for all users as well as for registered and non-registered users alone. Non-registered users use 0.7 more turns than registered users. The only difference between registered and non-registered users from the system's point of view is the presence of PPOI. We believe that the use of PPOI enables users to obtain the information they want efficiently with fewer dialog turns.

Task Type	**All**	**Registered**	**Non-registered**
Traffic	3.56	3.33	4.08
Gas Prices	3.73	3.54	4.14
Weather	3.80	3.61	4.41
Total	**3.65**	**3.44**	**4.14**

Table 3. Averaged number of turns per task type.

This theory is further validated when we examined the task completion rate. Figure 4 shows the task completion rates for the various tasks as a function of all users, registered and non-registered users. Overall, there is a 65.6% task completion rate. It is interesting to note, however, that registered users obtain a consistent task completion rate of about 70% across all tasks, while the task completion rate of non-registered users varies dramatically from 48% for the traffic task to 64% for the weather task. The traffic application is the only application that requires multiple locations: both an origin and destination. Coincidentally, traffic is also the application that is most likely to use PPOI as many users query the system for traffic information during their commutes between home and work. For calls made during these times, the registered users have only to confirm that they would like the traffic report between home and work, while non-registered users have to convey two locations to the system for the same request. Thus, the use of PPOI results in fewer turns in the dialog, and leads to a significantly higher task-completion rate for registered users.

5.2 User evaluation

To obtain a more subjective evaluation of the Commute UX system, we sent out a web-based survey to users of Commute UX who had made at least one call to the system and those who participated in the user study discussed in Section 4.2, whether they were registered or not. From this solicitation, we received 23 responses.

The survey asked the users to state their level of agreement to a series of statements, using a five-step scale that ranged from Strongly Agree to

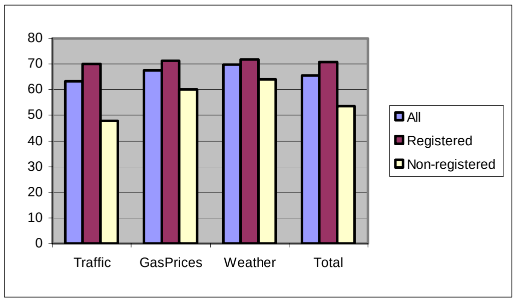

Figure 4. Completion ratio (%) per task.

Strongly Disagree. The questions and the responses are shown in Figure 5. As the results show, a majority of the respondents find the system useful and believe the system understands their speech. It is interesting to note that most users believe they are speaking in a natural manner, yet a similar number claim to only answer the questions the system asks. This contradicts our usual notion that system initiated dialog is not perceived as natural.

The other interesting conclusions from this data concern personalization. We note that several people use PPOI but most do not use PPOI other than the default "home" or "work" locations. Finally, we note that there are a significant number of users that always ask for the same information from the system. This indicates that there is a large opportunity for further improvement in task completion with additional personalization and user-specific grammar adaptation in this domain.

6 Discussion

In this paper, we presented a telephone dialog system for location-based services. It utilizes several key technologies for both recognizing and rendering spoken locations. We performed a user study to evaluate the users' response to various ways of describing a spoken location in terms of addresses, intersections, or points of interest, and designed our system to operate in the manner that both provided the best accuracy and was most preferred by users. The system also enables users to improve their experience with personal points of interest. The use of these personal locations resulted in dialogs with a higher task completion rate and fewer turns per task. A subjective user evaluation of the system revealed that most users had a positive experience with the system, but that there were opportunities for additional improvement through further personalization and user adaptation.

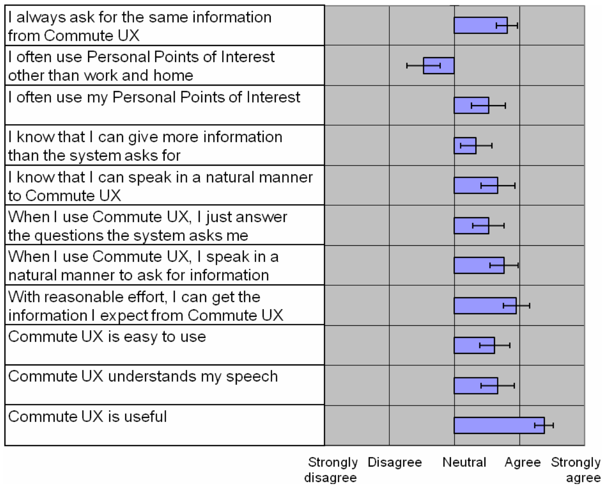

Figure 5. User survey results.

References

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, Vol. 22.

Bob Carpenter and Jennifer Chu-Carroll. 1998. Natural Language Call Routing: A Robust, Self-organizing Approach. *Proc. ICSLP*. Sydney, Australia

Alexander Gruenstein, Stephanie Seneff, and Chao Wang. 2006. Scalable and Portable Web-based Multimodal Dialogue Interaction with Geographical Databases. *Proc. ICSLP*, Pittsburgh, PA.

Hong-Kwang Jeff Kuo, Chin-Hui Lee, Imed Zitouni, Eric Fosler-Lussier, Egbert Ammicht 2002. Discriminative Training for Call Classification and Routing. Proc. ICSLP. Denver, CO.

Michael L. Seltzer, Yun-Cheng Ju, Ivan Tashev, and Alex Acero. 2007. Robust Location Understanding in Spoken Dialog Systems Using Intersections. Submitted to *Interspeech 2007*, Antwerp, Belgium.

Omer Tsimhoni, Daniel Smith, and Paul Green. 2004. Address entry while driving: speech recognition vs. a touch-screen keyboard. *Human Factors*, vol. 46, no. 4, pp. 600–610.

Anand Venkataraman, Horacio Franco, and Greg Myers. 2003. An architecture for rapid retrieval of structured information using speech with application to spoken address recognition. *Proc. of ASRU*, USVI.

Dong Yu, Yun-Cheng Ju, Ye-Yi Wang, and Alex Acero. 2006. N-gram based filler model for robust grammar authoring. *Proc. ICASSP*, Toulouse, France.