CONTEXT-SENSITIVE HELP FOR MULTIMODAL DIALOGUE

Helen Wright Hastie, Michael Johnston, Patrick Ehlen

AT&T Labs - Research
180 Park Ave, Florham Park, NJ 07932
{hhastie, johnston, ehlen}@research.att.com

ABSTRACT

Multimodal interfaces offer users unprecedented flexibility in choosing a style of interaction. However, users are frequently unaware of or forget shorter or more effective multimodal or pen-based commands. This paper describes a working help system that leverages the capabilities of a multimodal interface in order to provide targeted, unobtrusive, context-sensitive help. This Multimodal Help System guides the user to the most effective way to specify a request, providing transferable knowledge that can be used in future requests without repeatedly invoking the help system.

1. INTRODUCTION

Multimodal systems allow user input and/or system outputs over multiple communication modalities such as speech, pen, graphics, and gesture [1]. They can provide users with an unprecedented degree of flexibility in their inputs, allowing them to switch modes in order to overcome recognition errors [2, 3]. Multimodal systems are still a relatively new and unknown technology. Even though they do involve natural input modalities, these are not necessarily in familiar combinations or contexts such as drawing on a map and talking to it. Therefore, multimodal systems need to clarify their range of capabilities and thereby increase the user's ability to interact with the system effectively. This paper gives a breakdown of such a Multimodal Help System and its application in a mobile multimodal city guide, MATCH (Multimodal Access To City Help) [4].

One way to provide assistance to the user is for the system to take initiative and guide the user through a particular request step by step. For instance, to help the user choose a restaurant the system could ask first for a cuisine preference, then a price range, and then a preferred location. The problem with this approach is that the user remains dependent on the system, repeating the same multi-turn, system-initiative process for every such request. Another approach would be to present the user with an online help page. This, however, increases the cognitive load as the user must switch between the help page and the task in hand.

As an alternative solution, we have developed a working help system that provides *unobtrusive*, *context-sensitive*, *targeted* help which guides the user to the most effective way to specify a request and provides transferable knowledge which can be used for future requests without invoking the help system. This help system is *unobtrusive* as it is triggered only by the user, using speech or pen. It is also *context-sensitive*, being dependent on a combination of factors, including the dialogue context and the current visual context. Finally, this *targeted* help avoids the distraction of switching the user to a separate help display.

In addition, our help system leverages the capabilities of a *multimodal* interface and primes users to say words and perform gestures that the system understands well. Initial data collection and user studies, showed a high degree of variability in user success with the MATCH system [4]. Although users received an extensive tutorial at the beginning of each session, many users made limited use of pen-based and multimodal commands. Out of a corpus of 338 multimodal exchanges, only 19% were pen only (compared to 50% speech only and 28% multimodal), even though the sentence accuracy for pen was 64%, while for speech it was 36%. Users who had significant speech recognition performance problems did not switch modes and make use of alternative, more reliable strategies using pen or multimodal combinations. This is in keeping with previous observations in studies regarding multimodal error correction for dictation [3, 5], that also found that dealing with multimodal interfaces is not necessarily intuitive, especially when dealing with errors. In addition, these studies found that dictation correction accuracy increases when users switch modalities. Given these factors, for our conversational multimodal system we would like to familiarize users with the full range of available input methods and encourage them to make use of them. Towards this goal, the system's help messages are themselves multimodal. For example, if the system is explaining how to get information about a restaurant it will use one which is currently displayed as an example, and draw around it with electronic ink to identify it, while simultaneously providing spoken output explaining the kind of request that can be made.

Section 2 briefly describes the MATCH system and the multimodal architecture underlying it. Section 3 gives a breakdown of the system design and implementation of the help component, including a classification scheme for the dialog moves involved in multimodal help; their method of selection; and how the prompts are realized using synchronized speech and electronic ink. Section 4 concludes the paper.

Thanks to AT&T Labs and DARPA ITO (Contract No. MDA972-99-3-0003) for supporting this research. We would also like to thank Srinivas Bangalore, Amanda Stent, Gunaranjan Vasireddy, Marilyn Walker, Candy Kamm, and Mazin Rahim.

2. THE MATCH MULTIMODAL ARCHITECTURE

Fig. 1. MATCH running on Fujitsu PDA

Our multimodal system, MATCH [4], provides mobile access to restaurant and subway information for New York City, on a Fujitsu tablet (Figure 1). The majority of commands can be given by speech, by pen, or by a multimodal combination of speech and pen. For example, a user might ask to see cheap Italian restaurants in Chelsea by saying *show cheap italian restaurants in chelsea*, by circling an area on the map and saying *show cheap italian restaurants in this neighborhood*, or, in a noisy or public environment, by circling an area and writing *cheap* and *italian* (Figure 2). The system responds to user requests with coordinated multimodal outputs, combining dynamic graphics with synthetic speech. For example, if the user circles a group of restaurants on the display and says *tell me about these places*, the system will describe each restaurant in turn, and as each is discussed a graphical callout appears next to that restaurant providing the details.

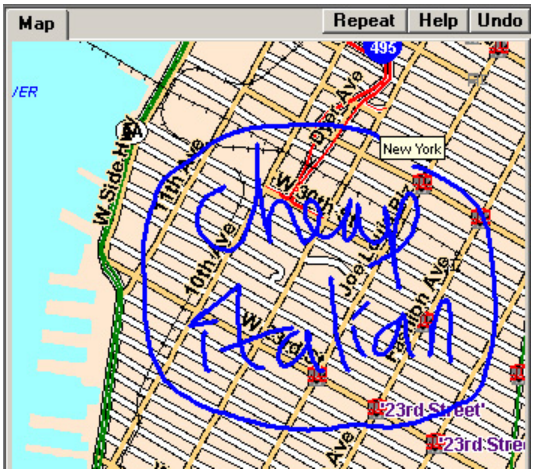

Fig. 2. Unimodal pen command

The multimodal architecture supporting MATCH consists of a series of agents which communicate through a facilitator, MCUBE (Figure 3). Users interact with the system

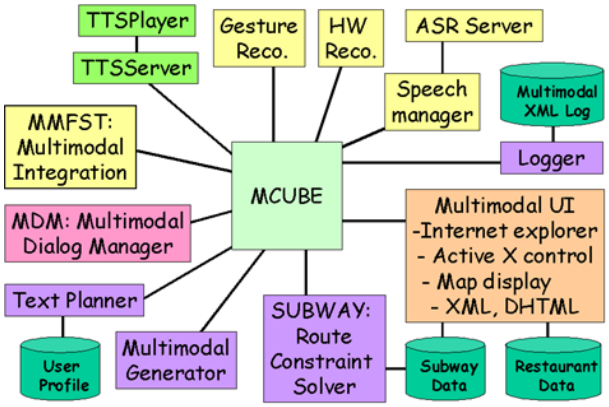

Fig. 3. Multimodal Architecture

through a Multimodal User Interface (UI) client (Figure 2). Their speech and ink are processed by speech recognition and handwriting/gesture recognition components respectively. The resulting lattices of potential words and gestures are integrated and assigned a meaning representation using a multimodal finite-state device [6, 4]. This provides an n-best list of potential interpretations of user inputs to a Multimodal Dialogue Manager (MDM, [4]), which are re-ranked in accordance with the current dialogue state. The command or query is passed back to the Multimodal UI which coordinates presentation of graphical content with TTS output from the TTS player. The Multimodal Help System comprises a module in the Multimodal Dialogue Manager outlined below and makes use of the Multimodal UI, described in Section 3.3.

2.1. MDM: Multimodal Dialogue Manager

The Multimodal Dialogue Manager is a speech-act based dialogue system inspired in part by the TrindiKit [7]. Each utterance is classified as a dialogue move which updates a common information state. The decision of which dialogue moves are to be performed by the system is made by the dialogue move engine. The dialogue manager consists of 3 specification files: an information state; resources and modules; and a control algorithm.

The information state contains a list of variable/binding pairs for different aspects of the dialogue, such as the history list and user and system intentions. The variables are defined prior to dialogue initiation but the bindings can be changed throughout the course of the dialogue. Resources are structured sources of information, for example the ontology resource defines structures such as the types of dialogue moves or the frame slots for command completion. Modules are groups of processes that perform similar functions. The information state is operated on by a control algorithm which calls one or more modules or resources. The top level dialogue flow controller has the following structure: it interprets the user's utterance, updates the information state, and selects the appropriate response and generates text for TTS.

There are certain aspects of this dialogue manager that are specialized for multimodal applications. For example,

the information state contains variables pertaining to the state of the visible map (such as the zoom level and the number of entities showing) that are updated via a message from the Multimodal UI every time the map changes. The system is predominantly user initiative, allowing the user to switch topics/commands freely. The system does take the initiative when a command cannot be completed due to missing information, though the user is free to start over with a new command.

3. MULTIMODAL HELP DESIGN AND IMPLEMENTATION

The Multimodal Help System is an independent module in the MDM which is only initiated when the user calls for help, presses the help button (Figure 2 top right), or writes *help* on the screen. It follows the same flow control as the main MDM: interpreting the type of help requested; updating the infostate; selecting the appropriate HELP DIALOGUE MOVES (HDM); generating text for TTS; and firing graphical actions. Certain information about help is held in the information state, such as the current help type, the help history, and hypotheses of the user's intentions for completing the current help exchange.

In the following section, we will discuss the dialogue strategy for help as captured by the set of HELP DIALOGUE MOVES. In section 3.2, we will discuss how a HELP DIALOGUE MOVE is chosen given the user's dialogue and the visual context. Finally in section 3.3, we illustrate how the system produces multimodal output for the chosen move.

3.1. Help Dialogue Moves

Dialogue act notation schemes for system prompts vary greatly among systems, depending on their dialogue strategy and domain. DATE (Dialogue Act Tagging for Evaluation) [8] provides a cross-system notation for system turns and does include meta-communication acts (known as SITUATION-FRAME ACTS). However, these notation systems do not address help strategy in any detail. Furthermore, these schemes have primarily been applied to spoken rather than multimodal dialogue. For the purpose of developing a HELP DIALOGUE MOVE scheme, we adopt a *shallow* approach, as accessing the help module of the Dialogue Manager should only occupy 2 exchanges at most. Our initial set of HELP DIALOGUE MOVES are domain specific, but in ongoing work we are abstracting from these to a more general set of help moves which can be re-used over a broad class of applications.

There are three main areas where the user may need help in this application: finding restaurants, getting subway directions, or moving round the map. As this is a multi-modal system, there are a number of ways the user can ask for help. They can either say "help," write it on the screen, or press the help button on the UI (see Figure 2). Alternatively, the request can be more specific, by saying, for example, *Help me find a restaurant* or *How do I move around the map*, or by writing *restaurant help* or *subway ?*.

The set of system HELP DIALOGUE MOVES is divided into queries and exemplary declaratives. If the user does not specify a particular type of help, and it is not clear from the context, the dialogue manager uses one of the QUERY HELP DIALOGUE MOVES: REST_HELP, INZONE_QUERY or INFOREQ_QUERY. Examples of these moves are given in the top part of Table 1. Otherwise, if the type of help the user requires is clear, one of the other moves is used based on helping the user in the three areas: RESTAURANTS, SUBWAY or MAP. These EXEMPLARY HELP DIALOGUE MOVES are given in Table 1.

3.2. Help Dialogue Move Selection Rules

One of the main goals of this help system is to be *context-sensitive*, in order to minimize diversion of the user from the task at hand. This is achieved through the application of the set of HDM rules in the help module of the MDM. These rules are triggered by the current context as represented in the information state. The diagram in Figure 4 depicts the context where the different HDMs are fired. This is based on both *visual context* and *dialogue history*. We will discuss each of these in turn.

The system uses various different properties of the visual context, including: the zoom level on the map and the number of entities (in this case restaurants) showing on the map. These attributes are stored in the information state and updated in accordance with messages that the UI sends when the map display changes. Zoom level provides a good example of the relevance of visual context in addition to dialog context. A user can ask MATCH to show entities at any zoom level on the map, as with, *show italian restaurants in Manhattan*. This request would result in a high zoom-level where entities appear as clusters of small dots, making it difficult to circle one and get information about a specific restaurant. So, if help is requested about selecting a restaurant, the user must be steered to a lower zoom level in this visual context. If we follow the right-hand side of Figure 4, we can see that the HDM INFOREQ_ZOOMOUT is called in this scenario. Example 4.1 in Figure 4 gives a relevant prompt and graphical action, where the user is directed to circle an area and write or say *zoom*. These graphical actions are described in greater detail in Section 3.3.

If the map is zoomed in and restaurants are showing then it is important to give help related to those restaurants on the screen. INFOREQ_ZOOMIN takes a restaurant on the screen as an example and shows the user how to get details about that restaurant. Example 4.2 in Figure 4 gives an INFOREQ_ZOOMIN prompt and associated action whereby a restaurant is circled and *phone* is written on the screen. Similarly, ZOOMIN_SUBWAY_HELP called in the same context takes an entity on the screen and uses it to exemplify how one can get subway directions to or from a particular place.

The dialogue context comes into play when we want to examine how much information the user has already established so as not to repeat an instruction that the user has already performed. For example, when giving instructions on getting subway directions, the system looks up in the dialogue history to see if the source of a route has been given (e.g., the user's current location). If the source is already established then the system helps the user select a destination (e.g., one of the restaurants on the screen) by firing DEST_HELP

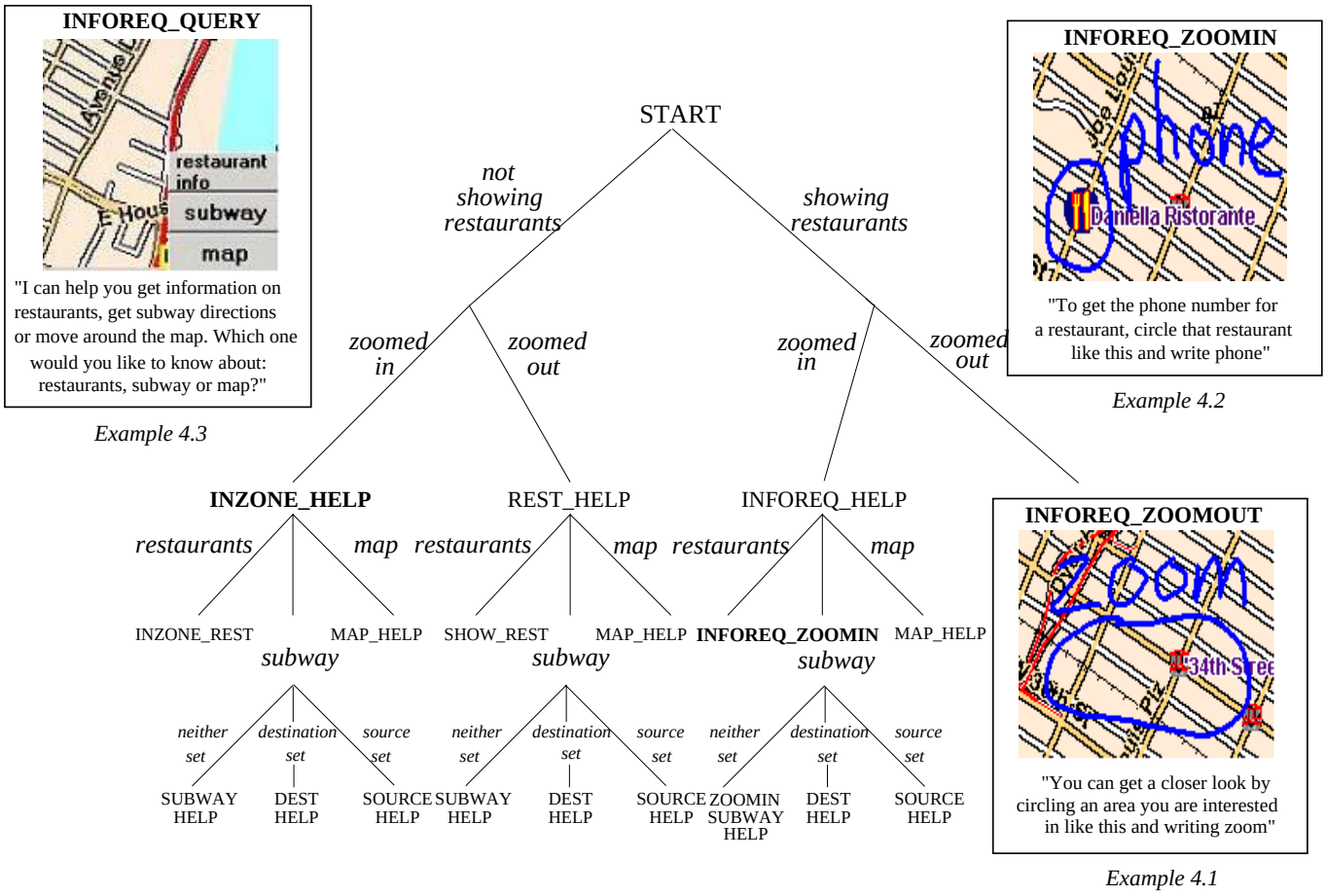

Fig. 4. Schema for firing HDMs based on visual and dialogue context

(see Table 1).

While it is possible to constrain help based on context, in a system of any complexity multiple kinds of help will be relevant in a particular context. For example, if the user has just successfully viewed restaurants and is at a sufficiently low zoom level, the system could either help the user with information about restaurants or help with subway directions to one of them. In this case, the MDM takes initiative with a QUERY HELP DIALOGUE MOVE and asks the user which kind of help they would like (e.g., INFOREQ_QUERY): *I can help you get information about a restaurant, get subway directions to one, or move around the map. Which one would you like to know about: restaurant information, subway or map?* The user is free to respond using speech or pen, but the system also provides a *triage widget* (list of buttons) in the bottom right of the display which allows the user to make a choice using a single, unambiguous pen tap (see example 4.3 in Figure 4). This widget is only available for a single turn, after which it disappears. It does not clutter the display and appears only in specific contexts where it is maximally useful. Buttons offer the user an error-free method of communicating with the system; particularly useful if poor ASR performance is contributing to the user's need for help.

3.3. Multimodal Help Output Generation

As described in the introduction, users should understand all the capabilities of the system, especially the modes of input available (speech, pen and handwriting). By employing all 3 modes in the help system, we hope to avoid priming the user into using only one mode. In addition, combining auditory and pictorial information can lighten the load on working memory resources, compared to tutorials where the auditory information is presented only as text and requires the learner to switch between text and picture to arrive at an integrated representation of the subject matter [9].

The MATCH architecture provides multimodal output capabilities. The Multimodal UI is capable of various graphical actions, including dynamically replaying words and gestures on the map and using ink to highlight specific entities. In addition, the UI has functionality to examine the current set of displayed restaurants and select one which is close to the center of the screen and does not have other restaurants nearby so it can be used to illustrate how to select a restaurant by circling it. The AT&T Natural Voices TTS engine is integrated into the architecture, and TTS prompts can be temporally synchronized with graphical actions in the UI.

Each of the HDMs has an associated graphical action or

set of actions (see Table 1). The query HDMs are associated with triage widgets (Section 3.2). The declarative, exemplary HDMs are associated with actions involving drawing gestures and words on the display. Examples of words and gestures associated with actions are given in Table 1. One such example is illustrated in Example 4.2 in Figure 4. An INFOREQ_ZOOMIN HDM is combined with SELECT_REST and WRITE_PHONE help actions, thus forming a MULTIMODAL HELP DIALOGUE MOVE (MHDM).

Synchronization between the spoken portion of a MHDM and the graphical actions is coordinated by the Multimodal UI. The MDM help messages to the UI consist of a prompt along with a series of specified graphical actions (Figure 5). The UI builds a stack of the graphical actions and sends the prompt to the TTS agent to be realized as synthetic speech. The prompt contains bookmarks which are returned by the TTS agent as those sections of the prompt are rendered. When these bookmarks are received the UI pops off the next action from the stack and executes it. For this task, we were able to synchronize speech and gesture effectively by placing the bookmarks a word or so back from the associated spoken phrase.

```
<mhdm>
  <speech>You can get information about
  restaurants such as the phone number,
  address or review. For example, to get
  the phone number circle a restaurant,
  <bookmark>1</bookmark> like this,
  and say or write <bookmark>2</bookmark>
  phone.</speech>
  <actions>
    <action><id>1</id>
      <spec>select(restaurant)</spec>
    </action>
    <action><id>2</id>
      <spec>writeink(phone)</spec>
    </action>
  </actions>
</mhdm>
```

Fig. 5. MULTIMODAL HELP DIALOGUE MOVE Specification

4. CONCLUSION

It remains a matter of some debate how effective online help can be for broad, open domain applications such as word processing and dictation. However, in interactive information access applications such as MATCH, the domain is more limited and the interaction more conversational in nature. The dialogue manager is able to tailor help far more tightly to the context, enabling help to be more targeted and unobtrusive, and therefore more likely to be effective.

Our approach to presenting targeted help differs from previous work on context-sensitive help for spoken dialogue in that it leverages the capabilities of a multimodal interface and utilizes visual context in addition to the dialogue context. Stein [10] examines the role of context-sensitive help in the SPEAK! multimodal information retrieval interface and compares textual help responses to spoken help responses. Our approach differs in the use of visual context to drive the help strategy and use of combinations of speech and pen gesture in help responses. [10] found that users preferred context-sensitive help and that spoken output allowed users to concentrate on the retrieval task.

In summary, we have presented a new and innovative approach to responding to users' requests for help in a multimodal dialogue system. The system maximizes the relevance of help by selecting HELP DIALOGUE MOVES on the basis of both the dialogue and visual context. The system utilizes synchronized multimodal output to provide help situated within the current visual context of the interface. This primes users to utilize the full range of input strategies made available by a multimodal interface and avoids distracting them from their conversational goals. Where context does not fully determine the appropriate help response, context-specific ephemeral interface widgets are employed to enable users to rapidly indicate their needs.

5. REFERENCES

[1] E. André, "Natural language in multimedia/multimodal systems," in *Handbook of Computational Linguistics*, Ruslan Mitkov, Ed. OUP, 2002.

[2] S. L. Oviatt and R. van Gent, "Error resolution during multimodal human-computer interaction," in *Proceedings of ICSLP-96, Philadelphia*, 1996.

[3] B. Suhm, B. Myers, and A. Waibel, "Model-based and empirical evaluation of multimodal interactive error correction," in *Proceedings of CHI*, 1999.

[4] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor, "MATCH: An architecture for multimodal dialog systems," in *Proceedings of ACL-02, Philadelphia*, 2002.

[5] C. Karat, C. Haverson, D. Horn, and J. Karat, "Patterns of entry and correction in large vocabulary continuous speech recognition systems," in *Proceedings of CHI*, 1999.

[6] M. Johnston and S. Bangalore, "Finite-state multimodal parsing and understanding," in *Proceedings of COLING 2000*, Saarbrücken, Germany, 2000.

[7] S. Larsson, P. Bohlin, J. Bos, and D. Traum, "TrindiKit manual," Tech. Rep., TRINDI Deliverable D2.2, 1999.

[8] M. Walker, R. Passonneau, and J. Boland, "Quantitative and qualitative evaluation of darpa communicator spoken dialogue systems," in *Proceedings of the 39rd Annual Meeting of the Association for Computational Linguistics (ACL/EACL-2001)*, 2001.

[9] R. Moreno and R. E. Mayer, "Cognitive principles of multimedia learning," *Journal of Educational Psychology*, vol. 91, pp. 358–368, 1999.

[10] Adelheit Stein, "Active Help and User Guidance in a Multimodal Information System: A Usability Study," in *Report ABIS-98, 6. Workshop Adaptivitt und Benutzermodellierung in interaktiven Softwaresystemen.*, U.J. Timm and M. Rössel, Eds. Erlangen: FORWISS Report, 1998.

QUERY HDM	**Example**	**HELP ACTION**	**Example**
REST_HELP	*I can help you find a restaurant, get subway directions or move around the map. Which one would you like to know about: restaurants, subway or map?*	TRIAGE_BUTTONS	
INZONE_HELP	*I can help you find a restaurant in this area, get subway directions or move around the map.*	TRIAGE_BUTTONS	
INFOREQ_HELP	*I can help you get information about a restaurant, get subway directions to one or move around the map.*	TRIAGE_BUTTONS	
EXEMPLARY HDM	**Example**	**HELP ACTION**	**Example**
SHOW_REST	*You can find restaurants based on price, food type and location. For example, you could say, show inexpensive italian restaurants in Chelsea. Alternatively, you could circle an area you are interested in, like this, and write italian and cheap.*	SELECT_AREA WRITE_REST	circle restaurant *cheap italian*
INZONE_REST	*I can give you information on restaurants in this area based their price and food type. For example, you could say, show expensive French restaurants. Alternatively, you could write expensive and French on the map.*	WRITE_REST	*expensive french*
INFOREQ_ZOOMIN	*You can get information about restaurants such as the phone number, address or review. For example, to get the phone number circle a restaurant, like this, and say or write phone.*	SELECT_REST WRITE_PHONE	circle restaurant *phone*
INFOREQ_ZOOMOUT	*You can get a closer look at restaurants by circling the area or the restaurants you're interested in and saying for example, zoom in here, or just writing zoom.*	SELECT_AREA WRITE_ZOOM	circle an area *zoom*
SUBWAY_HELP	*I can give you subway directions between any 2 points in New York City. For example you could say: how do I get to the cloisters? Alternatively, you could circle where you want to go and write direction or route.*	SELECT_AREA WRITE_SUBWAY	circle an area *route*
DEST_HELP	*The destination can be a neighborhood, a landmark, cross streets or a point on the map. For example, if want to go to penn station you could just say penn station, or write it on the map.*	WRITE_POI	*penn station*
SOURCE_HELP	*The source can be a neighborhood, a landmark, cross streets or a point on the map. For example, if you are at penn station you could just say penn station, or write it on the map.*	WRITE_POI	*penn station*
ZOOMIN_SUBWAY_HELP	*I can give you subway directions to one of these restaurants. For example, you could say, how do I get to the here, and circle the restaurant.*	SELECT_REST	circle restaurant
MAP_HELP	*I can help you move around the map. For example, you could say, show chelsea or show penn station or say zoom and circle an area on the map.*	SELECT_AREA WRITE_ZOOM	circle an area *zoom*

Table 1. Example Multimodal Help Dialogue Moves