

MULTIMODAL LANGUAGE PROCESSING FOR MOBILE INFORMATION ACCESS

Michael Johnston, Srinivas Bangalore, Amanda Stent, Gunaranjan Vasireddy, Patrick Ehlen

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

ABSTRACT

Interfaces for mobile information access need to allow users flexibility in their choice of modes and interaction style in accordance with their preferences, the task at hand, and their physical and social environment. This paper describes the approach to multimodal language processing in MATCH (Multimodal Access To City Help), a mobile multimodal speech-pen interface to restaurant and subway information for New York City. Finite-state methods for multimodal integration and understanding enable users to interact using pen, speech, or dynamic combinations of the two, and a speech-act based multimodal dialogue manager enables mixed-initiative multimodal dialogue.

1. LANGUAGE PROCESSING FOR MOBILE SYSTEMS

Mobile information access devices (PDAs, tablet PCs, next generation phones) offer limited screen real estate and no keyboard or mouse, making complex graphical interfaces cumbersome. Multimodal interfaces can address this problem by enabling speech and pen input and output combining synthetic speech and graphics (See [1] for a detailed overview of previous work on multimodal input and output). Furthermore, since mobile devices are used in situations involving different physical and social environments, tasks, and users, they need to allow users to provide input in whichever mode or combination of modes are most appropriate given the situation and the user's preferences.

Our testbed multimodal application MATCH (Multimodal Access To City Help) allows all commands to be expressed either by speech, by pen, or multimodally. This is achieved by capturing the parsing, integration, and understanding of speech and gesture inputs in a single multimodal grammar which is compiled into a multimodal finite-state device. This device is tightly integrated with a speech-act based multimodal dialog manager enabling users to complete commands either in a single turn or over the course of a number of dialogue turns. In Section 2 we describe the MATCH application. In Section 3, we describe the multimodal language processing architecture underlying MATCH.

2. THE MATCH APPLICATION

Urban environments present a complex and constantly changing body of information regarding restaurants, cinema and theatre schedules, transportation topology, and timetables. This information is most valuable if it can be delivered effectively while mobile, since users needs change while they are out and the information itself is dynamic (e.g. train times change and shows get cancelled).

Thanks to AT&T labs and DARPA ITO (contract No. MDA972-99-3-0003) for financial support.

MATCH is a working city guide and navigation system that enables mobile users to access restaurant and subway information for New York City (NYC). MATCH runs standalone on a Fujitsu pen computer, yet can also run in client-server mode across a wireless network. The user interacts with a graphical interface displaying restaurant listings and a dynamic map showing locations and street information (the Multimodal UI). They are free to give commands or reply to requests using speech, by drawing on the display with a stylus, or using synchronous multimodal combinations of the two modes. For example, they can request to see restaurants using the spoken command *show cheap italian restaurants in chelsea*. The system will then zoom to the appropriate map location and show the locations of restaurants on the map. Alternatively, they could give the same command multimodally by circling an area on the map and saying *show cheap italian restaurants in this neighborhood*. If the immediate environment is too noisy or public, the same command can be given completely in pen as in Figure 1, by circling an area and writing *cheap* and *italian*.



Fig. 1. Unimodal pen command

The user can ask for the review, cuisine, phone number, address, or other information for a restaurant or set of restaurants. The system responds with graphical callouts on the display, synchronized with synthetic speech output. For example, if the user says *phone numbers for these three restaurants* and circles a total of three restaurants as in Figure 2, the system will draw a callout with the restaurant name and number and say, for example *Le Zie can be reached at 212-206-8686*, for each restaurant in turn (Figure 3). These information seeking commands can also be issued solely with pen. For example, the user could alternatively have circled the restaurants and written *phone*.

The system also provides subway directions. For example, if the user says *How do I get to this place?* and circles one of the restaurants displayed on the map the system will ask *Where do you want to go from?* The user can then respond with speech e.g.: *25th*



Fig. 2. Two area gestures

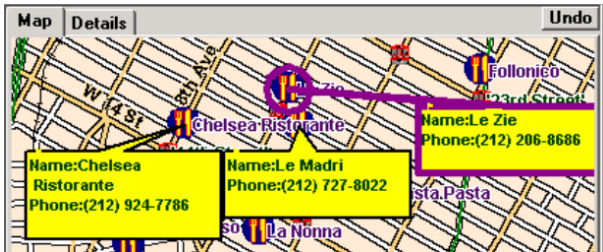


Fig. 3. Phone query callouts

Street and 3rd Avenue, with pen by writing e.g. 25th St & 3rd Ave, or multimodally: e.g. from here (with a circle gesture indicating the location). The system then calculates the optimal subway route and dynamically generates a multimodal presentation indicating the series of actions the user needs to take (Figure 4).

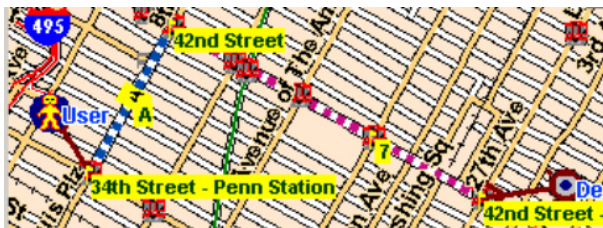


Fig. 4. Multimodal subway route

3. MULTIMODAL LANGUAGE PROCESSING

The multimodal architecture which supports MATCH consists of a series of agents which communicate through a Java-based facilitator MCUBE (Figure 5). We focus in this paper on multimodal input processing: the handling and representation of speech and electronic ink, their integration and interpretation, and the multimodal dialogue manager. [2] presents an experiment on text planning within the MATCH architecture. [3] describes the approach to mobile multimodal logging for MATCH.

3.1. Speech Input Handling

In order to provide spoken input, the user must hit a click-to-speak button on the Multimodal UI. We found that in an application such as MATCH which provides extensive unimodal pen-based interaction, it was preferable to use click-to-speak rather than pen-to-speak or open-mike. With pen-to-speak, spurious speech results received in noisy environments can disrupt unimodal pen commands. The click-to-speak button activates a speech manager running on the device which gathers audio and communicates with a recognition server (AT&T's Watson speech recognition engine).

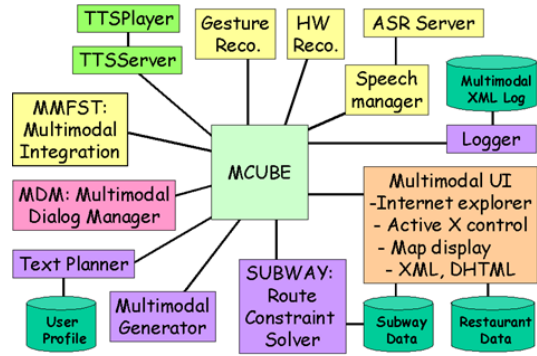


Fig. 5. MATCH Multimodal architecture

The output from the recognition server is a lattice of possible word string hypotheses with associated costs. This lattice is passed to the multimodal integrator (MMFST).

3.2. Recognizing and Representing Electronic Ink

Just as we determine a lattice of possible word strings for the audio signal in the speech mode, similarly for the gesture mode we need to generate a lattice of possible classifications and interpretations of the electronic ink. A given sequence of ink strokes may contain symbolic gestures such as lines and arrows, handwritten words, and selections of entities on the display.

When the user draws on the map, their ink is captured and any objects potentially selected, such as currently displayed restaurants or subway stations, are determined. The electronic ink is broken into a lattice of strokes and passed to the gesture recognition and handwriting recognition components to determine possible classifications of gestures and handwriting in the ink stream. Recognitions are performed both on individual strokes and combinations of strokes in the ink stroke lattice. For MATCH, the handwriting recognizer supports a vocabulary of 285 words, including attributes of restaurants (e.g. 'chinese', 'cheap') and zones and points of interest (e.g. 'soho', 'empire', 'state', 'building'). The gesture recognizer recognizes a set of 10 basic gestures, including lines, arrows, areas, points, and question marks. It uses a variant of Rubine's classic template-based gesture recognition algorithm [4] trained on a corpus of sample gestures. In addition to classifying gestures, the gesture recognition component also extracts features such as the base and head of arrows. The gesture and handwriting recognition components enrich the ink stroke lattice with possible classifications of strokes and stroke combinations, and pass this enriched stroke lattice back to the Multimodal UI. The Multimodal UI then takes this classified stroke lattice and the selection information and builds a lattice representation of all the possible interpretations of the user's ink, which it passes to MMFST.

3.2.1. Representation of Complex Pen-based Input

The representation of pen input in MATCH is significantly more involved than in our earlier approach to finite-state multimodal language processing [5, 6], in which the gestures were sequences of simple deictic references to people (G_p) or organizations (G_o). The interpretations of electronic ink are encoded as symbol complexes of the following form $G FORM MEANING (NUMBER TYPE) SEM$. $FORM$ indicates the physical form of the gesture and has values such as *area*, *point*, *line*, *arrow*. $MEANING$ indicates the meaning of that form; for example an *area* can be either a *loc(ation)*

or a *selection*). *NUMBER* and *TYPE* indicate the number of entities in a selection (*1,2,3, many*) and their type (*restaurant, theatre*). *SEM* is a place holder for the specific content of the gesture, such as the points that make up an area or the identifiers of objects in a selection (e.g. *id1, id2*). For example, if as in Figure 2, the user makes two area gestures, one around a single restaurant and the other around two restaurants, the resulting gesture lattice will be as in Figure 6. The first gesture (node numbers 0-7) is either a reference to a location (*loc.*) (0-3,7) or a reference to a restaurant (*sel.*) (0-2,4-7). The second (nodes 7-13,16) is either a reference to a location (7-10,16) or to a set of two restaurants (7-9,11-13,16). If the user says *show chinese restaurants in this neighborhood and this neighborhood*, the path containing the two locations (0-3,7-10,16) will be taken when this lattice is combined with speech in MMFST. If the user says *tell me about this place and these places*, then the path with the adjacent selections is taken (0-2,4-9,11-13,16).

3.2.2. Aggregation of Gestures

When multiple selection gestures are present, an aggregation technique is employed in order to overcome the problems with deictic plurals and numerals described in [7]. Aggregation augments the gesture lattice with aggregate gestures that result from combining adjacent selection gestures. This allows a deictic expression like *these three restaurants* to combine with two area gestures, one which selects one restaurant and the other two, as long as their sum is three. It avoids the need to enumerate in the multimodal grammar all of the different possible combinations of gestures deictic numeral phrases could combine with. In our example (Figure 6), the aggregation process applies to the two adjacent selections and adds a selection of three restaurants (0-2,4,14-16). If the speech is *tell me about these or phone numbers for these three restaurants* then the aggregate path (0-2,4,14-16) will be chosen.

3.3. Multimodal Integration (MMFST)

The multimodal integrator (MMFST) takes the speech lattice and the gesture interpretation lattice and builds a meaning lattice which captures the potential joint interpretations of the speech and gesture inputs. MMFST uses a system of intelligent timeouts to work out how long to wait when speech or gesture is received. These timeouts are kept very short by making them conditional on activity in the other input mode. MMFST is notified when the user has hit the click-to-speak button, when a speech result arrives, and whether or not the user is inking on the display. When a speech lattice arrives, if inking is in progress, MMFST waits for the gesture lattice, otherwise it applies a short timeout and treats the speech as unimodal. When a gesture lattice arrives, if the user has hit click-to-speak MMFST waits for the speech result to arrive, otherwise it applies a short timeout and treats the gesture as unimodal.

We use an extension of the finite-state approach to multimodal input processing proposed by Johnston and Bangalore [5, 6]. In this approach, possibilities for multimodal integration and understanding are captured in a three tape finite state device in which the first tape represents the speech stream (words), the second the gesture stream (gesture symbols), and the third their combined meaning (meaning symbols). In essence, this device takes the speech and gesture lattices as inputs, consumes them using the first two tapes, and writes out a meaning lattice using the third tape. The three tape FSA is simulated using two transducers: $G:W$, which is used to align speech and gesture, and $G_W:M$, which takes a composite alphabet of speech and gesture symbols as input and outputs meaning. The gesture lattice G and speech lattice W are composed

with $G:W$ and the result is factored into an FSA G_W which is composed with $G_W:M$ to derive the meaning lattice M (See [5, 6] for details). In addition to more complex representation of gesture and aggregation, we have also extended the approach with a more general method for abstracting over specific gestural content.

3.3.1. Abstracting Over Specific Gestural Content

In order to capture multimodal integration using finite-state methods, it is necessary to abstract over specific aspects of gestural content. For example, all the different possible sequences of coordinates that could occur in an area gesture cannot be encoded in the FST. Our previous approach [6], was to assign the specific content of gestures to a series of numbered variables $e1, e2, e3$. This limits the number of gestural inputs that can be handled to the number of variables used. If a large number are used, then the resulting multimodal finite-state device increases significantly in size. This becomes a significant problem with more complex pen-based input and when aggregation of gestures is considered (Section 3.2.2), since a new variable is needed for each aggregate combination of gestures. We have developed a generalized solution to this approach using the finite-state calculus which avoids these problems and reduces the size of the multimodal finite-state transducer. We represent the gestural interpretation lattice as a transducer $I:G$, where G are gesture symbols (including a reserved symbol *SEM*) and I contains both gesture symbols and the specific contents. I and G differ only in cases where the gesture symbol on G is *SEM*, in which case the corresponding I symbol is the specific interpretation. In order to carry out the multimodal composition with the $G:W$ and $G_W:M$ machines, a projection on the G output side of $I:G$ is used. In the multimodal FST, in any place where content needs to be copied from the gesture tape to the meaning tape an arc $eps:SEM:SEM$ is used. After composition we take a projection $G:M$ of the resulting $G_W:M$ machine (basically we factor out the speech (W) information). This is composed with the original $I:G$ in order to reincorporate the specific contents that had to be left out of the finite-state process ($I:G \circ G:M = I:M$). In order to read off the meaning we concatenate symbols from the M side. If the M symbol is *SEM* we instead take the I symbol for that arc.

3.3.2. Multimodal Grammars

The multimodal finite-state transducers used at runtime are compiled from a declarative multimodal context-free grammar which captures the structure and interpretation of multimodal and unimodal commands. This grammar captures not just multimodal integration patterns but also the parsing of speech and gesture, and the assignment of meaning. In Figure 7 we present a small simplified fragment capable of handling commands such as *phone numbers for these three restaurants*. A multimodal CFG differs from a normal CFG in that the terminals are triples: $W:G:M$, where W is the speech stream (words), G the gesture stream (gesture symbols), and M the meaning stream (meaning symbols). An XML representation for meaning is used to facilitate parsing and logging by other system components. The meaning tape symbols concatenate to form coherent XML expressions. The epsilon symbol (*eps*) indicates that a stream is empty in a given terminal.

Consider the example above where the user says *phone numbers for these three restaurants* and circles two groups of the restaurants (Figure 2). The gesture lattice Figure 6 is turned into a transducer $I:G$ with the same symbol on each side except for the *SEM* arcs which are split. For example, path 15-16 $SEM([id1, id2, id3])$ becomes $[id1, id2, id3]:SEM$. After G and the speech W are integrated using $G:W$ and $G_W:M$. The G path in the result is used to

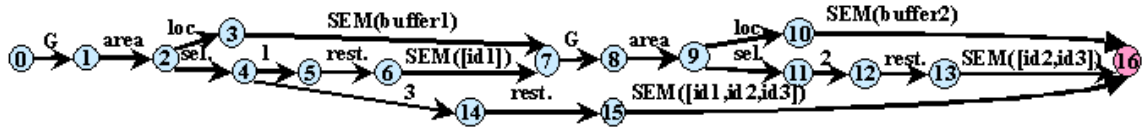


Fig. 6. Gesture lattice

S	→	eps:eps:<cmd> CMD eps:eps:</cmd>
CMD	→	phone:eps:<phone> numbers:eps:eps for:eps:eps DEICTICNP eps:eps:</phone>
DEICTICNP	→	DDETPL eps:area:eps eps:sel:eps NUM RESTPL eps:eps:<restaurant> eps:SEM:SEM eps:eps:</restaurant>
DDETPL	→	these:G:eps
RESTPL	→	restaurants:restaurant:eps
NUM	→	three:3:eps

Fig. 7. Multimodal grammar fragment

re-establish the connection between *SEM* symbols and their specific contents in $I:G$ ($I:G \circ G:M = I:M$). The meaning read off $I:M$ is $\langle cmd \rangle \langle phone \rangle \langle restaurant \rangle [id1, id2, id3] \langle /restaurant \rangle \langle /phone \rangle \langle /cmd \rangle$. This is passed to the multimodal dialog manager (MDM) and from there to the Multimodal UI where it results in the display in Figure 3 and coordinated TTS output.

3.4. Multimodal Dialog Manager (MDM)

The MDM is based on previous work on speech-act based models of dialog [8, 9]. It uses a Java-based toolkit (MDMKit) for writing dialog managers that embodies an approach similar to that used in TrindiKit [10]. It includes several rule-based processes that operate on a shared state. The state includes system and user intentions and beliefs, a dialog history and focus space, and information about the speaker, the domain, and the available modalities. The processes include an interpretation process, which selects the most likely interpretation of the user's input given the current state; an update process, which updates the state based on the selected interpretation; a selection process, which determines what the system's possible next moves are; and a generation process, which selects among the next moves and updates the system's model of the user's intentions as a result.

In the route query example in Section 2, MDM first receives a route query in which only the destination is specified *How do I get to this place?*. In the selection phase it consults the domain model and determines that a source is also required for a route. It adds a request to query the user for the source to the system's next moves. This move is selected and the generation process selects a prompt and sends it to the TTS component. The system asks *Where do you want to go from?*. If the user says or writes *25th Street and 3rd Avenue* then MMFSST will assign this input two possible interpretations. Either this is a request to zoom the display to the specified location or it is an assertion of a location. Since the MDM dialogue state indicates that it is waiting for an answer of the type location, MDM reranks the assertion as the most likely interpretation from the meaning lattice. A generalized overlay process [11] is used to take the content of the assertion (a location) and add it into the partial route request. The result is determined to be complete and passed on to the Multimodal UI, which uses the SUBWAY component, multimodal generator, and TTS to determine and present the optimal route to the user.

4. CONCLUSION

In MATCH, a single multimodal grammar captures the expression of commands in speech, pen, or multimodal combinations of the two. We have shown here how the finite-state approach to multimodal language processing can be extended to support representation of complex pen-based gestural input, generalized abstraction over specific gestural content, and aggregation of gestures. The finite-state multimodal integration component is integrated with a speech-act based multimodal dialog manager, enabling dialogue context to resolve ambiguous multimodal inputs, and allowing multimodal commands to be distributed over multiple dialogue turns. This approach gives users an unprecedented level of flexibility of interaction depending on their preferences, task, and physical and social environment.

5. REFERENCES

- [1] E. André, "Natural language in multimedia/multimodal systems," in *Handbook of Computational Linguistics*, Ruslan Mitkov, Ed. Oxford University Press, 2002.
- [2] A. Stent, M. Walker, S. Whittaker, and P. Maloor, "User-tailored generation for spoken dialogue: An experiment," in *Proceedings of ICSLP*, Denver, Colorado, 2002.
- [3] P. Ehlen, M. Johnston, and G. Vasireddy, "Collecting mobile multimodal data for MATCH," in *Proceedings of ICSLP*, Denver, Colorado, 2002.
- [4] D. Rubine, "Specifying gestures by example," *Computer graphics*, vol. 25, no. 4, pp. 329–337, 1991.
- [5] S. Bangalore and M. Johnston, "Tight-coupling of multimodal language processing with speech recognition," in *Proceedings of ICSLP*, Beijing, China, 2000.
- [6] M. Johnston and S. Bangalore, "Finite-state multimodal parsing and understanding," in *Proceedings of COLING*, Saarbrücken, Germany, 2000.
- [7] M. Johnston, "Deixis and conjunction in multimodal systems," in *Proceedings of COLING 2000*, Saarbrücken, Germany.
- [8] A. Stent, J. Dowding, J. Gawron, E. Bratt, and R. Moore, "The CommandTalk spoken dialogue system," in *Proceedings of ACL'99*, 1999.
- [9] C. Rich and C. Sidner, "COLLAGEN: A collaboration manager for software interface agents," *User Modeling and User-Adapted Interaction*, vol. 8, no. 3–4, pp. 315–350, 1998.
- [10] S. Larsson, P. Bohlin, J. Bos, and D. Traum, "Trindikit manual," Tech. Rep., TRINDI Deliverable D2.2, 1999.
- [11] J. Alexandersson and T. Becker, "Overlay as the basic operation for discourse processing in a multimodal dialogue system," in *2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 2001.