HOW FINITE STATE MACHINES CAN BE USED TO BUILD ERROR FREE MULTIMODAL INTERACTION SYSTEMS

Marie-Luce Bourguet
Queen Mary, University of
London. Mile End Road,
London, E1 4NS, UK
mlb@dcs.qmul.ac.uk

ABSTRACT

Recognition-based interaction technologies (e.g. speech and gesture recognition) are still error-prone. It has been shown that, in multimodal architectures, combining complementary input modes can contribute to automatic recovery from recognition errors. However, the degree to which error recovery can be achieved is dependent on the design of the interaction, i.e. on the set of multimodal constructions that the system is able to interpret. In this paper, we discuss several techniques, based on the finite state machine formalism, for modelling interaction designs and testing their robustness to recognition errors.

Keywords

multimodal interaction, recognition-based technologies, mutual disambiguation, error robustness, finite state machines, interaction design.

1. INTRODUCTION

Recent developments in recognition-based interaction technologies (e.g. speech and gesture recognition) open up a myriad of new possibilities for the design and implementation of multimodal user interfaces. However, designing and implementing systems that take advantage of these new interaction techniques is difficult. Recognition-based technologies are still error-prone; not only do recognisers make mistakes, but users' inputs are often ambiguous [3]. Speech recognition systems, for example, are sensitive to vocabulary size, quality of audio signal and variability of voice parameters. Current systems are extremely sensitive to background noise, and small changes in voice quality (due, for example, to the speaker having a cold) can significantly affect the performance of a recogniser even after the user has trained it. Also, the expressive power of speech is often obtained at the price of some ambiguity and imprecision in the messages.

In [5] it is shown that during the process of semantic fusion (a process in which several inputs in different modalities are integrated to form a semantically correct whole) multimodal architectures can achieve automatic recovery from recognition errors and false interpretations. During this integration process, complementary modalities (for example a speech utterance and a hand gesture) are combined to produce a message that contains less uncertainty and more meaning. The phenomenon in which an input signal in one modality allows recovery from recognition error or ambiguity in a second signal in a different modality is called *mutual disambiguation* of input modes [5].

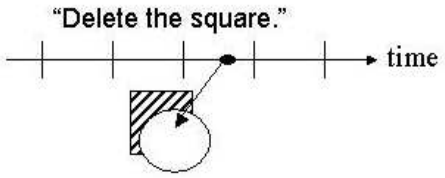

Figure 1: Example of mutual disambiguation

Figure 1 illustrates an example of a speech and pen command where both constituent inputs are incomplete or ambiguous when considered in isolation. The speech input ("Delete the square") is ambiguous if there is more than one square on the screen. Similarly, the pen input is ambiguous because of the two overlapping figures under the selection; it is also incomplete as it does not convey any meaningful action to apply on the selected object. However, the integration of these two ambiguous inputs yields a perfectly unambiguous message from which the action and the object of the action can both be unambiguously determined.

However, the degree to with which mutual disambiguation can operate in a given application is dependent on the

design of the interaction, i.e. on the set of multimodal constructions that the system is able to interpret. For example, for two easily confused words (from a speech recognition system point of view) such as "move" and "remove", mutual disambiguation will only operate if the two words can only be used in different types of multimodal constructions. If the speech input "remove" needs to be accompanied by a gesture drawn within the boundary of an object, whereas the speech input "move" needs to be accompanied by a dragging gesture that starts within an object and terminates outside the object, the fusion of the two inputs within the multimodal architecture will allow us to determine which speech input is more likely given the pen input (and vice versa). If, however, both speech inputs are equally likely given any pen input, mutual disambiguation will not work.

In this paper, we discuss a set of techniques for assessing the potential of different multimodal designs for mutual disambiguation of input signals. The goal of our research is to explore how simple prototyping tools and modelling techniques can be used to support the design of robust sets of multimodal commands. We show that finite state machines (FSMs) can be used to model different interaction designs, achieve modality integration and test error robustness.

2. INTERACTION DESIGNS

We define an interaction design as a description of how a particular set of modal user inputs can be used to activate a set of multimodal commands. For example, in one interaction design, the command "moving an object" may be specified by the following sequence of inputs: mouse-press on an object, mouse-drag and mouse-release. Alternatively, in another interaction design, the same command could be specified by a different sequence of inputs such as: mouse-click on an object, speech input "move" and mouse-click on a target position. Such sequences of inputs and actions can be modelled by simple finite state machines (FSMs).

Figure 2: Finite State Machine (FSM)

FSMs (see Figure 2) are a well-known technique for describing and controlling dialogs in graphical user interfaces [7]. According to [2], controlling a complete dialog with FSMs can present some significant drawbacks (such as promoting the use of modes), but FSMs are particularly appropriate for modelling dialogs at the command level. An FSM typically consists of states, events, transitions and actions. A transition has a source and a target state, and is executed when the FSM is in the source state and the event associated with the transition occurs. Upon the execution of a transition, an action associated with it can be triggered. Typically, a dialog starts with the machine being in a "start" state; as user inputs arrive, they are compared against the transitions leaving the current state. If the event matches the transition, the FSM moves to the state at the other end of the transition (target state); if no matching transition is found, the FSM usually moves to a special error state.

Figure 3 shows the representations of a "move" command and a "remove" command using the FSM formalism.

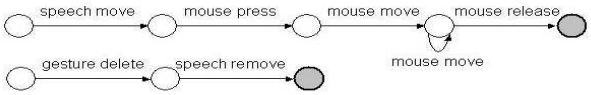

Figure 3: FSMs in which two different commands ("move" and "remove") are declared.

We have implemented a simple tool to help designers to declare and test multimodal designs based on the FSM formalism [1]. Figure 4 shows the global architecture of the system. The toolkit comprises two main modules: the "Interaction Model" and the "Multimodal Engine" to which is connected the core of the application (where the functions of the application are implemented).

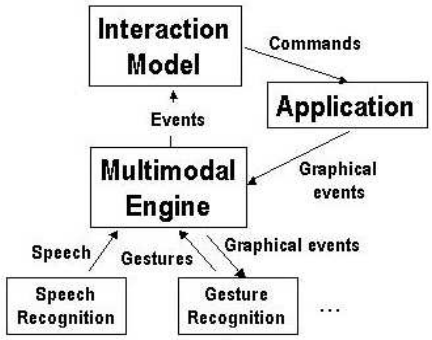

Figure 4: Toolkit global architecture.

The Interaction Model contains all the FSMs to be tested. The Multimodal Engine is responsible for controlling the different input channels and recognition systems (e.g. speech and gesture recognition systems) and for managing user inputs. Managing user inputs includes (1) formatting them so they can be matched with transition events in the interaction model, and (2) handling and dispatching recognition hypotheses as they are delivered by the recognition engines. To complement the toolkit, we have developed a graphical editor, which supports the task of designing and declaring interaction models. New interaction models can be created and existing ones can be uploaded for modification (iterative design). The declaration of an FSM is performed in a purely graphical way. New states and transitions can simply be drawn in the editor, while events and actions are typed or selected from a list provided with the corresponding application. Each interaction model is saved in an XML file and then reconstructed and implemented in Java, as described in [6], for subsequent use and testing with the Multimodal Engine.

3. TESTING ERROR ROBUSTNESS

As explained in the introduction, the design of an interaction model has some implications for the efficiency of mutual disambiguation and error robustness. In this section, we discuss how our FSM-based interaction-modelling framework can take advantage of the availability of multiple recognition hypotheses to handle uncertainty and test different degrees of robustness. We suggest several methods for the automatic correction of recognition errors: passive error handling, priority to first hypothesis, multiple dispatch, probabilistic FSM and backtracking.

3.1 Passive Error Handling

By simply not having an error state, an FSM can naturally filter out erroneous recognition hypotheses. The mis-recognised inputs are simply ignored, because they will not match the transition events of the current state,. The user may then choose to either repeat the input or reformulate it in order to increase the chances of good recognition. Once the input is correctly recognized, the dialog can resume.

Figure 3 shows an example where passive error handling is effective. Let us imagine that the user intends to move an object on the display; he or she says "move" but the machine understands "remove". Neither of the two FSMs can execute a transition so the speech input is ignored and the user can simply reiterate the speech. He or she may choose to say "delete" in order to avoid the previous confusion happening again. Both "move" and "delete" are assigned the tag [move] against which the transition event is compared.

Passive error handling is a very simple technique that does not achieve error correction but is able to filter out erroneous recognition results. It is appropriate for testing the robustness of simple interaction models, where all FSMs are significantly different from each other.

3.2 Priority to First Hypothesis

When the recognition engine delivers more than one recognition hypothesis, an alternative strategy can be used. In the previous example, we saw that, given the interaction model of Figure 3, the first hypothesis is ignored because it does not match any of the FSMs' transition events. The event handler may then dispatch the second hypothesis, which will be accepted by the first FSM. In this case, the user does not need to repeat the input, as the recognition error has been automatically corrected.

In order to work, this technique relies on the availability of several recognition hypotheses and on the fact that the correct speech input is present in one of these hypotheses. In the case of simple interaction models, it can be used to test the potential of a design for the automatic correction of recognition errors.

3.3 Multiple Dispatch

Another strategy is to dispatch the entire set of recognition hypotheses one after the other. Let us still imagine that the speech engine has delivered two recognition hypotheses: "remove" and "move", but let us now consider the interaction model shown in Figure 5 (note that the models shown in Figures 3 and 5 differ in that the bottom FSMs accept different inputs in their first transition). In the model shown in Figure 5, the bottom FSM will accept the first hypothesis ("remove") and the top one will accept the second hypothesis ("move"). Both FSMs will then reach their final state at the same time and the application will receive two concurrent actions and sets of inputs. The application will be in charge of deciding which of the two commands is the most probable. It is not aware of the order in which the speech recognition hypotheses were delivered, so for a more sophisticated strategy, the use of probabilistic FSMs is necessary.

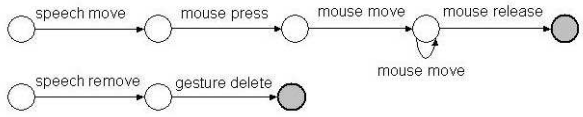

Figure 5: Robust interaction model with a "multiple dispatch" strategy

3.4 Probabilistic State Machines

The use of probabilistic state machines for dialog management for inputs with uncertainty has been discussed in [2]. This technique is relevant and applicable to multimodal interaction. The main difference between a probabilistic model and the traditional model of FSMs is that instead of having a single current state, a probabilistic FSM can have a distribution of alternative states. The probability that the machine is in any of these states is calculated based on a probability associated with each of the alternative user inputs. A drawback to this technique is that it relies on the recognition engines to provide a confidence score with each of the alternative hypotheses. Alternatively, it is suggested in [2] that when these confidence scores are not provided, they can be calculated a priori from a confusion matrix, built while the recognition system is being trained. However, sensitivity to the environment (e.g. background noise) that is characteristic of recognition technologies makes these confusion matrices highly unstable and hence unreliable. One potential advantage to this technique, though, is that the probability of a state that triggered an action can be communicated to the application. The application can then combine this probability with its internal models to evaluate if an action should be executed or not, or to compare several concurrent actions.

3.5 Backtracking

When an erroneous input has been accepted by an FSM, backtracking may be useful. It should be considered in the following situations:

- when the event to be matched is deterministic (e.g. a mouse press) but the user input cannot be accepted;
- when an action associated with a transition cannot be executed in the application (for example when the application cannot find an object under a mouse press).

In these situations, the FSM should backtrack to its previous state, to acknowledge that the user's intention does not match what is modelled in the FSM. Alternatively, the FSM could move to an error state.

Backtracking may also be considered when no transition has been executed, or when no user input has been received for a certain period of time. Temporal thresholds should be fixed to determine how long to wait for additional inputs [4].

4. DISCUSSION AND CONCLUSION

In this paper, we have suggested several techniques for error testing in the design phase of the production of a multimodal system. These techniques exhibit diverse levels of efficiency, complexity and scalability. The complexity of the interaction model to be tested, the availability of several recognition hypotheses and the desired level of robustness should indicate the use of one technique over another.

The robustness of the interaction mainly depends on the complexity of the interaction model and how efficiently mutual disambiguation can operate within that model. The simplest error handling methods (passive error handling and priority to first hypothesis) are unlikely to work for an interaction model consisting of many FSMs, as it becomes highly probable that the first hypothesis will be accepted by at least one of the FSMs. An interaction model for which such techniques can efficiently achieve error correction can be considered as very robust.

As the interaction models become more complex, more sophisticated error handling methods must be deployed (e.g. probabilistic FSMs and backtracking). However, these methods require that the application is able to handle uncertain actions. In this case, interaction robustness not only depends on the design of the interaction model but also on the ability of the application to reason about concurrent actions.

The iterative design, implementation and testing of multimodal user interfaces is difficult, due to a lack of supporting tools for designers and developers. In response to this, we have developed a toolkit that aims to facilitate this process. In particular, modality integration, error handling and user input management are handled by the toolkit in a transparent manner. We have also developed a graphical tool that facilitates the process of declaring interaction models (see section 3). However, in the near future we are planning to automatically generate interaction models from experimental observations. Potential users will be asked to freely produce actions with the aim of activating specific multimodal commands. These actions will then form the basis for the automatic generation of FSMs. These automatically generated FSMs will then be tested for error robustness using the techniques presented in this paper.

5. ACKNOWLEDGEMENT

This research is supported by the Nuffield Foundation under grant NUF-NAL 00.

6. REFERENCES

[1] Bourguet M.L. (2002). A Toolkit for Creating and Testing Multimodal Interface Designs, *in companion proceedings of UIST'02*, 29-30.

[2] Hudson S. & Newell G (1992). Probabilistic Sate Machines: Dialog Management for Inputs with Uncertainty, *in Proceedings of UIST'92*, 199-208.

[3] Mankoff J., Hudson S. & Abowd G. (2000). Providing integrated toolkit-level support for ambiguity in recognition-based interfaces, *in Proceedings of CHI'00*, ACM Press, 368-375.

[4] Oviatt S., De Angeli A. & Kuhn K. (1997). Integration and synchronisation of input modes during multimodal human-computer interaction, *in Proceedings of CHI'97*, ACM Press, 415-422.

[5] Oviatt S. (2000). Taming recognition errors with a multimodal interface, *in Communications of the ACM*, **43 (9)**, ACM Press, 45-51.

[6] Van Gurp J. & Bosch J. (1999). On the implementation of finite state machines, *in Proceedings of the IASTED 3rd International Conference on Software Engineering and Applications*.

[7] Wasserman A. (1985). Extending State Transition Diagrams for the Specification of Human-Computer Interaction, *IEEE Transactions on Software Engineering*, **11 (8)**, 699-713.