Multi-threaded Interaction Management for Dynamic Spatial Applications

Srinivasan Janarthanam
Interaction Lab
Heriot-Watt University
Edinburgh
sc445@hw.ac.uk

Oliver Lemon
Interaction Lab
Heriot-Watt University
Edinburgh
o.lemon@hw.ac.uk

Abstract

We present a multi-threaded Interaction Manager (IM) that is used to track different dimensions of user-system conversations that are required to interleave with each other in a coherent and timely manner. This is explained in the context of a spoken dialogue system for pedestrian navigation and city question-answering, with information push about nearby or visible points-of-interest (PoI).

1 Introduction

We present a multi-threaded Interaction Manager (IM) that is used to track different dimensions of user-system conversations and interleave the different converational threads coherently. The IM that we present interacts with the user in a spatial domain and interleaves navigation information along with historical and cultural information about the entities that users can see around them. In addition, it aims to answer questions that users might have about those entities. This presents a complex conversational situation where several conversational threads have to be interleaved in such a way that the system utterances are presented to the user at the right time but in a prioritised order, and with bridging utterances when threads are interrupted and resumed. For instance, a navigation instruction may be important (since the user is walking up to a junction at which they need to turn) and therefore it needs to be spoken before continuing information presentation about an entity or answering other ongoing questions.

2 Related work

Previously, multi-threaded interaction was used to handle multiple simultaneous tasks in human-robot interaction (HRI) scenarios (Lemon and Gruenstein, 2004). This idea also turns out to be important for cases where humans are interacting with a variety of different web-services in parallel. Human multitasking in dialogue is discussed in (Yang et al., 2008).

(Lemon and Gruenstein, 2004) presented a multi-threaded dialogue management approach for managing several concurrent tasks in an HRI scenario. The robot could, for example be flying to a location while simultaneously searching for a vehicle, and utterances about both tasks could be interleaved. Here, conversational threads were managed using a representation called the "Dialogue Move Tree", which represented conversational threads as branches of the tree, linked to an "Activity Tree" which represented the states of ongoing robot tasks (deliver medical supplies, fly to a waypoint, search for a truck), which could be active simultaneously. The situation for our pedestrian navigation and information system is similar - concurrent tasks need to be managed coherently via conversation. The approach adopted in this paper is similar to (Lemon and Gruenstein, 2004). However, in this work we separate out a domain-general thread called 'dialogue control' which handles generic issues like clarification of reference across all tasks. This increasing modularisation of the dialogue threads makes it possible to learn individual dialogue policies for each one, in future work.

(Nakano et al., 2008) presented an approach where one of the several expert modules handling different tasks is activated based on the user input, but only one verbal expert is active at any one time. In contrast to this, we present an approach where several thread managers each handling a different task can be activated in parallel and their outputs stored and retrieved based on priority.

3 Multi-threaded IM

The Interaction Manager (IM) is the central component of any spoken dialogue system architec-

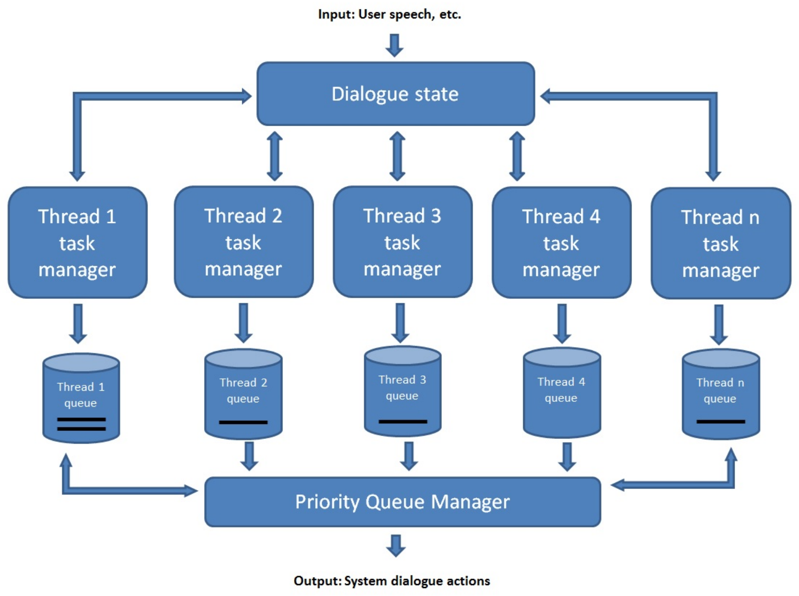

Figure 1: Interaction Manager Architecture

ture. Generally, it takes as input the user's utterances in the form of dialogue acts from the parser and identifies the next dialogue action to present to the user. Dialogue about a domain task is managed using a dialogue strategy or policy (e.g. (Young, 2000; Lemon and Pietquin, 2007)). A dialogue policy is a mapping between dialogue states and dialogue actions, which are semantic representations of what the system should say next.

In order to handle multiple tasks simultaneously, we present an architecture for a multi-threaded interaction manager that treats conversation about each domain task as a thread. These conversational threads are interleaved and managed using techniques such as multi-queuing, priority based pushing, and queue revision. We describe these techniques below. The architecture of the Interaction Manager is shown in figure 1.

Multi-threading and queuing

In order to manage complex interactions involving several conversational tasks/topics, we propose that the each task be handled by a thread manager within the interaction management framework. Each such manager will handle a conversational thread using a dialogue policy. Each thread manager will be fed with the input from the user and the dialogue actions generated will be stored in separate queues. This approach allows the interaction manager to produce several dialogue actions at the same time although for different conversational tasks.

Prioritised Queue Management

Dialogue actions from the several threads are stored in separate queues. The queues can be assigned priorities that decide the order in which items from the queues will be popped. The dialogue actions in the queues are pushed to the user based on an order of priority (see below). This priority can either be fixed or dynamic based on context. The system and user engagement should also be checked so that system utterances are pushed only when the system and user are not speaking already.

Queue Revision: resuming and bridging

The dialogue actions are generated and stored in queues. Therefore, there is a difference between the time they are generated and time that they are pushed. Therefore dialogue actions in the queues are revised periodically to reflect changes in context. Obsolete dialogue actions will have to removed for two reasons. Firstly, pushing them to the user may make the conversation incoherent because the system may be speaking about an entity that is no longer relevant and secondly, these obsolete dialogue actions may delay other other important dialogue actions from being pushed on time. In addition, it may also be useful to edit the dialogue actions to include discourse markers to signify topic change (Yang et al., 2008) and bridge phrases to reintroduce a previous topic. We discuss some examples later in section 4.3.

4 SPACEBOOK Interaction Manager

As a part of the SpaceBook EU FP7 project, we implemented the above design for a multi-threaded interaction manager that presents the user with navigational instructions, pushes PoI information, and manages QA questions (Janarthanam et al., 2013). It receives the user's input in the form of a dialogue act (DA) from the ASR module and the user's location (latitude and longitude), orientation, and speed from the Pedestrian Tracker module. Based on these inputs and the dialogue context, the IM responds with a system output dialogue act. It should be noted that the location coordinates of the user are sent to the IM every 2 seconds. This allows the IM to generate location aware information at a high frequency. In addition, the IM has to deal with incoming requests and responses from the user's spoken inputs. With the possibility of system utterances being generated at a frequency of one every two seconds, there is a need for an efficient mechanism to manage the conversation and reduce the risk of overloading the user with information. These tasks are treated as separate conversational threads.

4.1 Conversational Threads

The SpaceBook IM manages the conversation using five conversational threads using dedicated task managers. Three threads: 'navigation', 'question answering' and 'PoI pushing', represent the core tasks of our system. In addition, for handling the issues in dialogue management, we introduce two threads: 'dialogue control' and 'request response'. These different threads represent the state of different dimensions of the user-system conversation that need to interleave with each other coherently. Each of the threads is managed by a thread manager using a dialogue policy. Each thread can generate a dialogue action depending on the context, as described below:

Dialogue Control
During the course of the conversation, the IM uses this thread to manage user requests for repetition, issues with unparsed (i.e. not understood) user utterances, utterances that have low ASR confidence, and so on. The dialogue control thread is also used to manage reference resolution in cases where referring expressions are underspecified. The IM resolves anaphoric references by keeping a record of entities mentioned in the dialogue context. It stores the name and type information for each entity (such as landmark, building, etc) mentioned in previous utterances by either user or system. Subsequent user references to these entities using expressions such as "the museum", "the cafe", and so on, are resolved by searching for the latest entity of the given type. In cases where the IM cannot resolve the referent, it asks the user to clarify.

Request Response
The user can also initiate tasks that interest him/her at anytime during the conversation. These tasks include searching for an entity (e.g. a museum or a restaurant), requesting navigation instructions to a destination, and asking questions about the entities in the city database such as their location ("Where is X?", "How far is X?"). During navigation, users might want to ask questions about the destination, ask for next instructions, etc. All these user requests are handled using the request response thread. For instance, when the user asks for directions, the IM resolves the destination entity (perhaps using clarification) in the city model and acknowledges the user request. The task is then further handled using the Navigation thread.

Navigation
The IM identifies the location of the destination entity and queries a city database (Bartie and Mackaness, 2013) for a route plan. Using the route plan, the navigation thread presents step-by-step instructions to the user based on the current location and orientation of the user. The IM continuously monitors users to determine if at any time they are deviating from the planned route and provides corrective instructions. As users get near to the next node on the route plan, the next instruction is given. The IM uses highly salient visible landmarks (Bartie et al., 2013) and popular landmarks near the nodes to instruct the user (e.g. "When you reach Clydesdale Bank, turn left on to Nicolson Square"). The IM also informs users when they pass by recognisable landmarks, just to reassure them that they are on the right track (e.g. "You will pass by Tesco on the right"). When the user is close to his/her destination, the IM determines whether the destination is visible to the user, informs the user, and closes the task.

Usr: I want to go to the National Gallery.
Sys: Ok. I am now looking for directions to the National Gallery.
Sys: The National Gallery is on Queens Street at about 300 meters away.
....
Sys: Can you see the Bank of Scotland about 300 meters away on your left? It has a big green dome.
Usr: Yes. I can see it.
Sys: Walk towards it.
Sys: Can you see the National Gallery in front of you at about 100 meters? It is a building with Roman architecture.
Usr: Yes.
Sys: You have now reached your destination.

Question Answering
The system also answers ad hoc questions from the user (e.g. "Who is David Hume?", "What is the Old College?", "Who was William Wallace", etc). These are sent to the QA server and answered based on responses from the Question-Answering (QA) server (Janarthanam et al., 2013). The dialogue policy here is to answer the user's question with the first snippet available and ask the user to request for more if more snippets are available and he or she is interested.

Pushing PoI Information
When the user is mobile, the IM identifies popular points of interest (PoI) on the route based on two factors: proximity and visibility. The dialogue policy is to introduce the PoI, query the QA server for snippets and push the first snippet to the user. The user is encouraged to ask for more information if he/she is interested.

Sys: In front of you, about 200 meters away is Old College. It has a grey dome on top.
Sys: Situated on South Bridge, Old College is . . .
Sys: Ask for more information if interested.

4.2 Priority assignment in SpaceBook

Priority is assigned to the above dialogue threads as follows:

Priority 1. Dialogue control (repeat request, clarifications etc)
Priority 2. Responding to user requests
Priority 3. System initiated navigation task actions
Priority 4. Responses to User-initiated QA actions
Priority 5. PoI Push actions

For instance, informing the user of a PoI could be delayed if the user needs to be given an instruction to turn at the junction he is approaching.

4.3 Queue revision and bridging utterances

The queues need to be revised at regular intervals in order to keep the information in them relevant to context. For instance, the dialogue action of informing the user of his/her location is deleted after 5 seconds, as this tends to become obsolete. Similarly, dialogue actions corresponding to information segments in PoI and QA queues are edited to inform the utterance generator of other intervening dialogue actions so that it can use appropriate bridge phrases to reintroduce the focus of the conversational thread. For instance, as shown in the example below, the utterance generator inserts a bridge phrase (i.e. "More on Old College") to reintroduce the focus of the PoI push task because of the intervening user request and the subsequent system response.

Sys: In front of you, about 200 meters away is the Old College. It has a grey dome on top.
User: Where am I?
Sys: You are on Chambers street.
Sys: **More on Old College**. Situated on South Bridge, the Old College is......

5 Conclusion

We presented an architecture for a multi-threaded Interaction Manager that can handle multiple conversational tasks. We also described an implementation of the architecture in a dynamic spatial environment. The SpaceBook IM is a multi-tasking IM that aims to interleave navigation information along with historical information about the entities users can see around them. In addition, it aims to answer questions users might have about those entities.

Acknowledgements
The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 270019 (SPACEBOOK project `www.spacebook-project.org`).

References

P. Bartie and W. Mackaness. 2013. D3.1.2: The SpaceBook City Model. Technical report, The SPACEBOOK Project (FP7/2011-2014 grant agreement no. 270019).

P. Bartie, W. Mackaness, M. Fredriksson, and J. Konigsmann. 2013. D2.1.2 Final Viewshed Component. Technical report, The SPACEBOOK Project (FP7/2011-2014 grant agreement no. 270019).

S. Janarthanam, O. Lemon, P. Bartie, T. Dalmas, A. Dickinson, X. Liu, W. Mackaness, and B. Webber. 2013. Evaluating a city exploration dialogue system combining question-answering and pedestrian navigation. In *Proc. ACL 2013*.

Oliver Lemon and Alexander Gruenstein. 2004. Multithreaded context for robust conversational interfaces: context-sensitive speech recognition and interpretation of corrective fragments. *ACM Transactions on Computer-Human Interaction (ACM TOCHI)*, 11(3):241– 267.

Oliver Lemon and Olivier Pietquin. 2007. Machine learning for spoken dialogue systems. In *Interspeech*.

Mikio Nakano, Kotaro Funakoshi, Yuji Hasegawa, and Hiroshi Tsujino. 2008. A framework for building conversational agents based on a multi-expert model. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, SIGdial '08, pages 88–91, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fan Yang, Peter A. Heeman, and Andrew Kun. 2008. Switching to real-time tasks in multi-tasking dialogue. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 1025–1032, Stroudsburg, PA, USA. Association for Computational Linguistics.

Steve Young. 2000. Probabilistic methods in spoken dialogue systems. *Philosophical Transactions of the Royal Society (Series A)*, 358(1769):1389–1402.