Utterance Planning in an Agent-based Dialogue System

Paul Thompson, Mark Stairmand and William Black
Department of Computation
UMIST, PO Box 88, Manchester, M60 1QD
{thompson,m.stairmand,wjb}@co.umist.ac.uk

This paper describes the Response Planning and Generation components of the Athos framework implemented in the DUMAS project. DUMAS investigates adaptive multilingual interaction techniques to handle both spoken and text input and to provide coordinated linguistic responses to the user [1, 3]. The components we describe for response planning and generation functions are implemented as Jaspis agents [2].

AthosMail was developed in the DUMAS project. This is a multilingual speech-based application which allows a user to access his/her mailbox using a telephone. Another application under development, AthosNews, will provide a speech interface to newspapers for visually impaired users, and provide both browse and search functionality.

1 Jaspis overview

Jaspis is a framework for adaptive speech applications, designed to support distributed, context-sensitive applications that adapt to the user and the environment. It is designed with multilingual applications in mind. The Jaspis architecture supports distributed coordinated components, shared system knowledge and system-level adaptation. Jaspis is characterised by a shared information store, evaluators, and small, stateless agents. The information store can be viewed as a shared blackboard to which all Jaspis components have access and contains, for example, the dialogue state and the user model. By exploiting the shared information store Jaspis components can be stateless; because information about dialogue state is not stored internally by the agents, the system is free to select any Jaspis agent as the most appropriate to react to a dialogue situation.

2 Response Planning and abstract utterances

The generation phase starts when pragmatic analysis of an utterance has been carried out by a Jaspis agent called PDTree. This process includes anchoring, discourse modelling and interpretation of the user's goal. The goal specifies how the system can meet the user's needs; it consists of a goal type, a query summary and a solution list. Having determined the goal the system must then generate an appropriate response.

The Response Planner (RP) produces an abstract plan for an utterance that can be realised in different ways by the output generator. It takes as input the goals and propositional content produced by PDTree, which is stored in the central Information Store. The Response Planner is based on (i) declarative rules that reference events, objects and relations between them (ii) a model of events, object and relations populated by propositional input from PDTree, described above.

The model is implemented as a set of generic relational database tables for events, objects and relations, and Java objects for domain specific properties (e.g. 'from' attribute of an email). The rules are unified with the dialogue model to establish which actions are appropriate to the current dialogue state. The primary function of the response planner is to select the appropriate utterance plan for the current dialogue situation, and to populate the plan accordingly.

2.1 Utterance plans

Utterance plans consist of three main parts. The PRED element indicates the type of system utterance that is to be produced. The value of this element, together with the current value of the explicity parameter are used to determine the sequence of utterance pieces that will form the system response. The DESCRIPTION, encodes any selection criteria specified in the preceding user query, whilst the MEMBERS element provides information about the items that match these criteria

Templates encapsulate states for which specific pre-determined utterance plans are appropriate. A template is a set of declarative rules, some with associated conditions, which are unified with the dialogue state recorded in the database tables. When the RP is invoked, it will establish whether the current state matches that encapsulated in any template, in which case the associated utterance plan is deemed appropriate. The Response Planner achieves this by attempting to unify each template in turn with the recorded dialogue state.

```
<RULESET>
   [DG_OBJ(_C, 'MAIL', _T, _G, _S, _P)]
   [DG_EVT(_E, _, T, G, S, P) {0,}
   [DG_REL(E, _, C, T, G, S, P) {0}]]
   [DG_SOL(_V, G, T) {1,}]
</RULESET>
```

Figure 1: A response template

The RP generates XML to specify the appropriate utterance plan. This output will reference variables matched in the unification process, and it is often necessary to group together values matched over the whole result set. A metalanguage within the template allows manipulation and grouping of values found in matching dialogue states.

In our AthosMail domain we are looking for such entities as 'mail' objects, 'person' objects, 'person objects playing a dative role', 'read' events. Our Newspaper application domain deals with different objects.

3 Generation of utterance plans

The section illustates how the Response Planner would produce an abstract utterance plan in response to the user query *Do I have any messages from Bob?*. The system will respond to such a query by enumerating the details of the messages that match the specified criteria.

The template shown in Figure 1 matches the dialogue state when the above utterance has been analysed. It specifies that it can handle the situation where there is a mail object referenced in a goal, with one or more solutions. It also stipulates that if there is an event specified in the same proposition as the mail object, then there must be no relation between the event and the mail object recorded. The utterance plan outline shown in Figure 2 is associated with this template.

The Response Planner analyses this plan and carries out the directives specified by the metalanguage, based on the variables bound during unification, to produce the fully populated abstract utterance plan. The next section describes how the system utterance is built from this plan.

```
%T (
   [PRED text=enumerate]
   [TYPE ]
   [ARGO
     [SET
       [DESCRIPTION
         [ATTVAL attribute='instanceOf' value='message']
         [ATTVAL attribute='status' value='{C.status}']
         [ATTVAL attribute='from'value='{C.from}']]
      [MEMBERS %C(
         [MEMBER
            [ATTVAL attribute='instanceOf' value='message']
            [ATTVAL attribute='index' value='$C.index$']] )]]] )
```

Figure 2: An utterance plan outline

4 Utterance generation

The generator agent takes as its input the abstract representation produced by the RP. Its output is a surface realisation of the utterance that is passed to the speech synthesizer to be read to the user. System responses are generated by concatenating sequences of pre-defined utterance segments. Some of these segments are fixed stings whilst parts of others are dynamically generated, e.g. by accessing information about messages that have been retrieved according to a user query.

A user modelling component produces a value for an explicity parameter (varying from 1 to 3), based on such factors as the user's experience with the system and their interaction style. The value of this parameter determines the sequence of utterance segments to use to produce the system response. The higher the value, the more verbose the system response. The abstract representation may be used as a basis for the generation of responses in languages other than English. Work is currently being undertaken to configure the generator to produce responses in Swedish.

The fully populated abstract utterance plan, following on from the example shown in Figure 2, is illustrated in Figure 3. The contents of the MEMBERS element show that 2 messages have been retrieved in response to the query. The indices given are the positions of these messages within the mailbox section of the Information Store, which facilitate retrieval of further information about the messages.

If an explicity value of 2 is assumed, the generator will first retrieve and concatenate together the appropriate set of utterance segments for an

```
 <ACTION>
  <PRED>enumerate</PRED>
  <ARGO>
    <SET>
       <DESCRIPTION>
          <ATTVAL attribute='instanceOf' value='message'></ATTVAL>
          <ATTVAL attribute='from' value='Bob'></ATTVAL>
       </DESCRIPTION>
       <MEMBERS>
         <MEMBER>
           <ATTVAL attribute='instanceOf' value='message'></ATTVAL>
           <ATTVAL attribute='index' value='2'></ATTVAL>
         </MEMBER>
         <MEMBER>
           <ATTVAL attribute='instanceOf' value='message'></ATTVAL>
           <ATTVAL attribute='index' value='5'></ATTVAL>
         </MEMBER>
       </MEMBERS>
     </SET>
   </ARGO>
</ACTION>
```

Figure 3: A fully populated abstract utterance plan

"enumerate" response, resulting in the following string:

> *There are [quantity] [message_description]. [message_details]. What would you like to do?*

The parts of the string in square brackets are variables, whose values are generated according to the current contents of the DESCRIPTION and MEMBERS elements of the ACTION. The [quantity] variable gets replaced by the number of messages retrieved in response to the user query, found by counting the number of MEMBER elements within MEMBERS, in this case 2. The [message_description] variable should "echo back" the user-specified selection criteria, found within the DESCRIPTION element. The value of the [message_details] variable gives information about each of the retrieved messages in the MEMBERS element, namely the values of the fields that were not specified in the user query. In this case, the value required for the "from" field was specified by the user, causing only the subject and date to be retrieved for each of the two messages. Below is an example of a response generated from the above ACTION:

> *There are 2 messages from Bob. Message 1 dated 1st June 2003*

with subject AthosMail and message 2 dated today with subject lunch. What would you like to do?

References

[1] Kristiina Jokinen and Björn Gambäck. DUMAS:Adaptation and Robust Information Processing for Mobile Speech Interfaces. In *Human Language Technology: The Baltic Perspective*, pages 115–120, Riga, Latvia, April 2004.

[2] Markku Turunen and Jaakko Hakulinen. Jaspis 2 - An Architecture for Supporting Distributed Spoken Dialogues. In *Proceedings of the 8th European Conference on Speech Communication and Technology*, pages 1913–1916, Geneva, Switzerland, September 2003. ISCA.

[3] Markku Turunen, Esa-Pekka Salonen, Mikko Hartikainen, Jaakko Hakulinen, Bill Black, Allan Ramsay, Adam Funk, Andrew Conroy, Paul Thompson, Mark Stairmand, Kristiina Jokinen, Jyrki Rissanen, Kari Kanto, Antti Kerminen, Björn Gambäck, Maria Cheadle, Fredrik Olsson, and Magnus Sahlgren. Athosmail - A Multilingual Adaptive Spoken Dialogue System for E-Mail Domain. In *Proceedings of the 20th International Conference on Computational Linguistics, Geneva, Switzerland. ACL*, August 2004. To appear in 'The Workshop on Robust and Adaptive Information Processing for Mobile Speech Interfaces'.