An XPath-based Discourse Analysis Module for Spoken Dialogue Systems

Giuseppe Di Fabbrizio
AT&T Labs - Research
180 Park Avenue
Florham Park, NJ 07932 – USA
pino@research.att.com

Charles Lewis
AT&T Labs- Research
180 Park Avenue
Florham Park, NJ 07932 – USA
clewis@research.att.com

ABSTRACT

This paper describes an XPath-based discourse analysis module for Spoken Dialogue Systems that allows the dialogue author to easily manipulate and query both the user input's semantic representation and the dialogue context using a simple and compact formalism. We show that, in managing the human-machine interaction, the discourse context and the dialogue history are effectively represented as Document Object Model (DOM) structures. DOM defines interfaces that dialogue scripts can use to dynamically access and update the content, the structure and the style of the documents. In general, this approach applies also to richer multimedia and multimodal interactions where the interpretation of the user input depends on a combination of input modalities.

Categories & Subject Descriptors: I.2.1 [**Application and Expert Systems**]: Natural Language Interfaces - I.2.7 [**Application and Expert Systems**]: Speech Recognition and Synthesis

General Terms: Design, Experimentation.

Keywords: Spoken Dialogue Systems, Discourse Analysis, XPath.

1. INTRODUCTION

Dialogue management systems are useful to simulate the process of dialogue between two or more agents (either humans, machines or a mix of both) exchanging information to achieve a specific goal. Accordingly to [1], a general dialogue system consists of three fundamental pattern recognition components: 1) a *semantic parser*, which includes an automatic speech recognition (ASR) module to convert the user's utterances into text and a *sentence interpretation* module that parses the sentences into semantic representation; 2) a *discourse analysis* module which derives the new dialogue context based on the previously executed states; and 3) a *dialogue manager* that iterates the possible responses based on a predefined dialogue strategy. The framework in [1] is formalized as a maximum a posteriori (MAP) [5] decision problem where the goal is to minimize the cost of reaching the final state or agents' objective. The clear separation of the previously defined modules 2) and 3) is missing in many existing architectures where the two are merged together as an interactive manager [4] or a generic dialogue manager module. We will show that maintaining this distinction improves the overall flexibility of the system.

In this paper we focus on the discourse analysis (DA) module. DA maps the semantic input representation for the specific dialogue domain to changes in the appropriate context variables. This provides the DM with the information it needs to decide the next step in its strategy for fulfilling the goals of the dialogue.

WWW 2004, May 17-22, 2004, New York, NY, USA.
ACM 1-58113-912-8/04/0005.

In a Natural Language Spoken Dialogue Systems, the DM provides the means for human-machine interaction. The DM helps the user to achieve the task that the dialogue is designed to support. It keeps track of the discourse context, and, generally, adapts the level of initiative to the user's skills. The sentence interpretation is provided by a spoken language understanding (SLU) module that extracts predefined call-types from the user input. The SLU also parses out information on common named entities such as phone numbers, dates, zip codes and others. The discourse analysis module merges the sentence interpretation into the discourse context, resolves ellipsis and anaphora references, and detects inconsistencies, ambiguities, and partial information based on dialogue history. In our system, these complex tasks are performed by evaluating the local and global contexts where state variables and dialogue execution trees are stored in a DOM document. An XPath [6] based interpreter models the DOM document as a tree of nodes and operates on the string-values to update or extract the needed values. In the next sections we will describe in detail the system architecture, show how the model drives the dialogue flow and illustrate this process with examples of disambiguation and reference resolution.

2. SYSTEM ARCHITECTURE

Figure 1 depicts the high level system architecture. The converted spoken speech into text is the input to the sentence interpretation module (SLU) that classifies the input and translates it to an XML semantic representation. Either the Natural Language Semantic Markup Language [7] (NLSML) or the Extensible MultiModal Annotation [8] (EMMA) are suitable formats. These markup languages are intended for use by systems that provide semantic interpretations for a variety of inputs, including, but not necessarily limited to, speech and natural language text input. We adopted a simplified version of NLSML where the classified call-types are enumerated with confidence scores in the tag `<class>` and the named entities are reported directly in the `<instance>` tag instead of the optional XForm data model. To achieve the classification task, we use an extended version of a boosting-style classification algorithm [2] trained on an application dependent corpus and combined with context-free grammars (CFGs) [3] for named entity extraction.

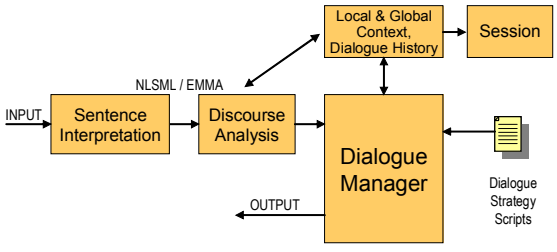

Figure 1. System Architecture

We assume that the sentence interpretation module attempts to extract the semantic interpretation of the utterance without considering the current context of the dialogue execution. The generated NLSML

document is fed to the discourse analysis module which updates the local context and the dialogue history DOM structures with the new input. Logical predicates, described by XPath expressions, allow the DA to rank classes and assign a contextual interpretation to the input based on the current context content. The local context stores the state variables and keeps track of the history for dialogue backtracking and repairs. The DM algorithm uses Augmented Transition Networks [9] (ATNs) to represent the dialogue flow. The ATN operates on the current input and the local context to control the interaction flow. State transitions are triggered by XPath expressions and generate the output actions when traversed. Finally, the session component enables session persistence, replication and failover support in a distributed environment.

3. Interaction with DOM and XPath

The dialogue context and the history are stored in a DOM tree where nodes contain variables or NLSMS sub-trees. Figure 2 shows a

```
<result>
 <interpretation>
  <input>I'm trying to find out what you have as far
as our account goes how does it stand </input>
  <instance>I'm trying to find out what you have as
far as our account goes how does it stand</instance>
 <classes>
 <class name="Tell(Account_Status)" score="0.96"/>
 <class name="Tell(Account_Features)" score="0.90"/>
 </classes>
 </interpretation>
</result>
```

Figure 2. Ambiguous Input

simplified NLSML tree for an ambiguous user utterance. The classification returns two confusable call-types, Tell(Account_Status) and Tell(Account_Features), with high confidence (>0.7) and close scores (<0.2). This condition is easily captured by the XPath expression `expr1: <cond expr1 ="(count(//class[@score > $threshold ]) = 2) and (((//class [position() = 1]/@score - //class [position() = 2] /@score < $closeScore))" expr2="true"/>`. Notice that we used the operator $ as compact way to dereference string-values of variables stored in the DOM nodes. $threshold, for example, is the string-value

```
<history>
 <turns>
  <turn type="concrete">
   <result>
   <interpretation>
    <input>could you tell me the repair department phone
number</input>
   <instance>could you tell me the <dept>REPAIR</dept>
phone number</instance>
  <classes>
   <class name="Req(Number)" score="0.94"/>
  </classes>
  </interpretation>
  </result>
  </turn>

  <turn type="discourse">
   <result> ...
    <instance>hello</instance> ...
  </turn>

  <turn type="command">
   <result>
    <interpretation>
    <input>could you repeat please</input>
    <instance>could you repeat that please</instance>
     <classes>
       <class name="Repeat" score="0.98"/>
     </classes>
    </interpretation>
   </result>
  </turn>
  </turns>
</history>
```

Figure 3. Dialogue history with anaphoric reference

of the node /local/threshold where the current value is set by the dialogue strategy script: `<var name="threshold" value="0.7"/>`. The DM will trigger the transition if this condition evaluates to true and will then engage a subsequent question to disambiguate the user's request, for example: *do you want your account status or the account features?*

Figure 3 is a simplified snapshot of the dialogue history where three turns are stored within the tag `<turns>`. In the last turn, the user is asking the system to repeat the repairs department phone number announced in the first turn, but the information to repeat is not explicitly stated (anaphora reference). The DA has to go back in the history of the dialogue and find the turn that the user is referring to. This translates to the following XPath expression: `<set name="repeat" expr="//turn[last() and @type = 'concrete']//class [@score>0.7]/@name"/>`. The attribute *type* specifies the category of the call-type in terms of user's intentions. Typically call-types can be categorized as *concrete*, when they covey information that contribute to the user request, as opposite to *discourse* or *command* types where the contribution is irrelevant to focus of the dialogue. Going back to the first concrete call-type request allows the DA to extract the reference correctly and assign the Req(Number) call type to the `repeat` context variable.

4. SUMMARY

The XPath standard has proven to be a powerful tool for the discourse analysis layer. This module must provide a level of processing distinct from the parsing-oriented SLU layer and the logic-oriented DM layer. The discourse analysis layer must act as a bridge between these two layers of processing, and speak both of their languages. In the system that we have described, XPath was able to analyze the output of the SLU to any degree of specificity required, and to communicate results into the variables used by the DM for decision making. Also, and as importantly, the XPath-based Discourse Analysis module was able to interact with both other layers without taxing its expressive power. XPath is a natural choice for this role.

5. REFERENCES

[1] X. Huang, A. Acero and H-W. Hon, Spoken Language Processing: A Guide to Theory, Algorithm and System Development, Prentice Hall PTR, NJ, 2001, pages 853-855.

[2] R. E. Schapire. The boosting approach to machine learning: An overview. In Proceedings of the MSRI Workshop on Nonlinear Estimation and Classification, Berkeley, CA, March, 2001.

[3] J. E. Hopcroft and J. D. Ullman: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979).

[4] W3C Multimodal Interaction Framework, W3C NOTE 06 May 2003, http://www.w3.org/TR/mmi-framework.

[5] L.Bahl, J.Jelinek, J.Raviv, and F.Raviv, Optimal Decoding of Linear Codes for minimising symbol error rate, IEEE Trans on Information Theory, vol. IT-20, pp.284-287, Mar 1974.

[6] XML Path Language (XPath), Version 1.0, W3C Recommendation 16 Nov 1999, http://www.w3.org/TR/xpath

[7] Natural Language Semantics Markup Language for the Speech Interface Framework, W3C Working Draft 20 Nov 2000, http://www.w3.org/TR/2000/WD-nl-spec-20001120

[8] EMMA: Extensible MultiModal Annotation markup language, W3C Working Draft 18 December 2003, http://www.w3.org/TR/ 2003/ WD-emma-20031218.

[9] D. Bobrow and B. Fraser. An augmented state transition network analysis procedure. In Proceedings of the IJCAI, pages 557-567,Washington, D.C, May, 1969.