Dialogue Management for Interactive Question Answering

Sanda M. Harabagiu, Marius Paşca and V. Finley Lăcătuşu
Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX 75275-0122
{sanda,mars,finley}@seas.smu.edu

Abstract

With the increasing availability of on-line information, either on the Internet or within Intranets, the need for Question Answering (Q&A) systems that allow user interaction expressed in natural language has become critical. A major obstacle in building robust, user-friendly Q&A systems is the need to enable a conversation with the user in which clarifications, follow-up questions and context specification are made possible. This paper presents INTERLOCUTOR, a generic interactive shell that performs dialogue management for open-domain Q&A systems through a set of dialogue strategies enabled by task templates.

Introduction

The recent explosion of on-line documents has determined a new, compelling framework for finding information that closely matches user needs: *Open-Domain Textual Question Answering*. Due to the fact that both questions and answers are expressed in natural language, Question and Answering (Q&A) methodologies deal with language ambiguities and incorporate Natural Language Processing (NLP) techniques. Several current NLP-based techniques are able to provide the framework of Open-Domain Q&A, i.e. answering questions in a manner that is independent of any specific domain, by extracting answers from large collections of texts. Typically, these systems combine semantic information brought forward by named entity recognizers with information derived from text passages retrieved by using question keywords.

Ideally, open-domain textual Q&A systems should incorporate semantic information pertaining to any domain, thus enabling the resolution of most ambiguities. Since at present such resources do not exist, ambiguities can be eliminated by *negotiating* the meaning of a question through a dialogue between the user and the Q&A system. In fact, the absence of an interactive component represents one of the major *knowledge engineering* bottlenecks in current Q&A systems. Instead of generating erroneous answers due to the incorrect interpretation of an ambiguous question or due to lack of sufficient information when extracting the answer, a Q&A system could benefit from a *generic interactive shell*, designed to manage a dialogue with the user.

In this paper we present INTERLOCUTOR, a generic interactive shell that (1) is used in conjunction with any Q&A system; (2) leverages the open-domain processing capabilities of a Q&A system; (3) provides additional information to the Q&A system, because it carries out a dialogue with the user. Furthermore in INTERLOCUTOR the dialogue management is based on information seeking motivations rather than presuppositions of the user's goals. Thus far, the best-performing and most robust dialogue processing systems have operated on sufficiently limited domains (e.g. queries about train schedules (Allen et al.1995), reading emails (Walker et al.1998-1) or battlefield simulations (Stent et al.1999)). Because of the limited domain, the dialogue systems could presume most of the user's goals and tailor accordingly the dialogue initiatives. The novelty of INTERLOCUTOR is that it abstracts away from domain-based dialogue management mechanisms by exploiting the semantic knowledge derived by the Q&A system and using it to generate dialogue strategies independently of the dialogue domain. Moreover, the INTERLOCUTOR framework uniformly addresses the problem of *mixed initiative interaction* between a user and a Q&A system by both generating clarification questions and interpreting follow-up questions and answers.

The rest of the paper is organized as follows. Section 2 details the architecture of the INTERLOCUTOR dialogue system whereas Section 3 presents our methodology of generating open-domain dialogue templates. Section 4 presents an algebra of dialogue templates whereas Section 5 defines the dialogue motivators. Section 6 presents some experimental results and Section 7 summarizes the conclusions.

Interactive Question Answering

Recent advances in NLP have made it possible to develop dialogue systems for many applications. Typically, the dialogue is managed by processing handcrafted knowledge templates that model the characteristic events associated with the application. This

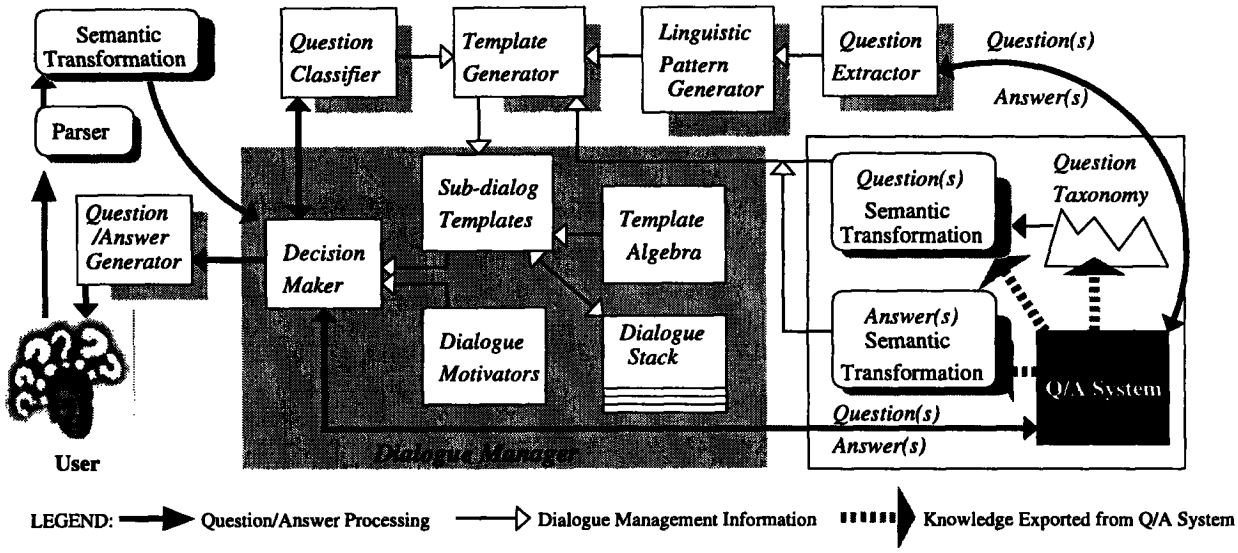

Figure 1: The Architecture of INTERLOCUTOR

framework is not viable in an interaction with an open-domain Q&A system. With INTERLOCUTOR we propose an alternative dialogue architecture, illustrated in Figure 1. Before submitting a question directly to the Q&A system, the question is parsed and transformed into a semantic representation that captures the relevant concepts and their interconnections. This representation is used to determine (1) the question complexity; and (2) the relationship to the previous dialogue. If the question is new or unrelated to the prior utterances, it is presented to the Q&A system, which returns a set of answers. Otherwise, before being submitted to the Q&A system, since the question may be the user's answer to a question generated by INTERLOCUTOR, it is first grounded in the existing dialogue, and a new question, containing also the contextual information is sent to the Q&A system. The question complexity is used to build a template of the information-seeking task. Each interaction fills a new template, that is organized into a hierarchical structure by the operators defined in a *Template Algebra* similar to the one reported in (Abella and Gorin 1999). This organization enables an analytical dialogue management, capable of handling complex strategies. A decision maker associates actions with the *dialogue motivators*, thus generating follow-up questions or presenting information to the user. The dialogue motivators provide the decision maker with the dialogue strategies that govern its behavior. The *Template Algebra* provides the building blocks needed to create new dialogue motivators and analyze them. For example, one of the dialogue motivators detects when a topic switch has occurred, and pushes all sub-dialogue templates on a stack.

There is also a second usage of the Q&A system in INTERLOCUTOR. This time, the Q&A system is not employed to find answers to a user's question, but rather it is used to provide information that enables the automatic generation of task templates. The information derived by processing the answers does not always contain everything related to a typical event for a given problem. Therefore, the collection needs to be mined for additional, related information. To mine the text, we first generate linguistic patterns typical for the domain of interest. Currently, scenario-level patterns for information extraction can be generated automatically if textual information relevant to the domain is available. The Q&A system retrieves the text passages that can be used to extract new information, that is encoded in task templates.

The role of the dialogue manager is to interact in a natural way to help the user complete the task of finding information through Q&A. Generally, previous research has treated the specification of the dialogue management strategy as an interactive design problem: several versions of a dialogue system are created, dialogue corpora are collected with human users interacting with different versions of the system, a number of evaluation metrics are collected for each dialogue, and the different versions are statistically compared. Due to the fact that in INTERLOCUTOR the task is not known beforehand, these methods cannot be applied. Instead we chose to design the dialogue manager by relying on a collection of *dialogue motivators* that are applied to the dialogue templates. The *dialogue manager algorithm* defined in INTERLOCUTOR is:

---

repeat
  for *all dialogue motivators* $\{DM\}$
    if DM_i *applies to template* T
      perform $Action(DM_i, T)$ *and obtain a template* T_A *from the user's answer.*
      Combine T *and* T_A *into* T *in Template Algebra.*
until *no dialogue motivator applies;*

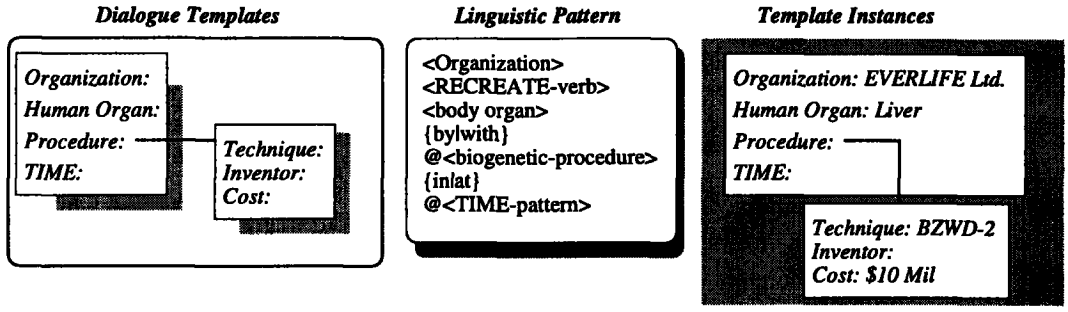

Figure 2: Examples of Dialogue Templates and some of their Instances.

Open-Domain Dialogue Templates

Open-domain dialogue templates can be generated by combining four knowledge sources: (1) the semantic transformations of the user's question; (2) the semantic transformations of the answers returned by the Q&A system; (3) a question taxonomy employed by the Q&A system, that is going to be ported to INTERLOCUTOR along with the semantic transformations; and (4) information indicating the degree of complexity of a question. Most Q&A systems process a question by first producing its parse, and then, based on the resulting dependency structures, generate a semantic representation. They also employ question taxonomies - therefore such standard resources can be made available to INTERLOCUTOR. Moreover we take into account the fact that a Q&A system may be employed by users with different degrees of sophistication. In order to better understand the nature of the Q&A task and to put this into perspective, we offer the following taxonomy:

• *Class 1:* Factual questions, with the answer found in a text snippet. The processing involves recognition of named entities, appositions and uses bag-of-words approaches.

Example:

Q: What is the largest city in Germany ?

A: ... George Bush visited Berlin, the largest city in Germany ...

• *Class 2:* Information extraction type of questions, for which the answer is found in multiple text snippets, scattered throughout a document and even across documents.

Example:

Q: What is the U.S position on human cloning ?

A: U.S. bans any forms of human cloning. Recent advances in biogenetics enable organ recreation

• *Class 3:* The question addresses a problem that needs to be summarized detected and summarized from several documents.

Example:

Q: What are the arguments for and against prayer in school ?

A: a summary generated from different sources

• *Class 4:* The processing of the question and the extraction of the answer rely on extensive domain knowledge.

Example:

Q: Should the Fed raise interest rates at their next meeting ?

A: If the Fed will raise the interest rates, the market will slow down. Recently analysts complained about the risk of inflation.

• *Class 5:* The question requires reasoning by analogy or other advanced reasoning mechanisms, developed in association with high-performance knowledge bases.

Example:

Q: What should be the US foreign policy in the Balkans now?

A: The US troops stationed in the Balkans are the guarantee for the peace-keeping process.

Because the dialogue templates model the Q&A task, their generation depends on the degree of the Q&A complexity. The template generation methodology differs for each complexity class. The generation of open-domain dialogue templates is described by the following steps:

□*Step 1: If the complexity class of the question = 1* select all the concepts directly connected to the answer type in the semantic representation of the question and its corresponding answer and label the slots of the template with their WordNet semantic class.

□*Step 2: If the complexity class of the question = 2* generate linguistic patterns to extract information from the *same* documents as the answers. Use this information to define new slots in the template and label them with their WordNet semantic class.

□*Step 3: If the complexity class of the question = 3* generate linguistic patterns to extract information from *different* documents than those containing the answers. Use this information to define new slots in the template and label them with their WordNet semantic class.

□*Step 4: If the complexity class of the question = 4* generate causality patterns that entail information related to the answer type. Use this information to define new slots in the template and label them with their WordNet semantic class.

□*Step 5: If the complexity class of the question = 5* use axiomatic information available from knowledge bases to determine reasoning patterns. For each reasoning

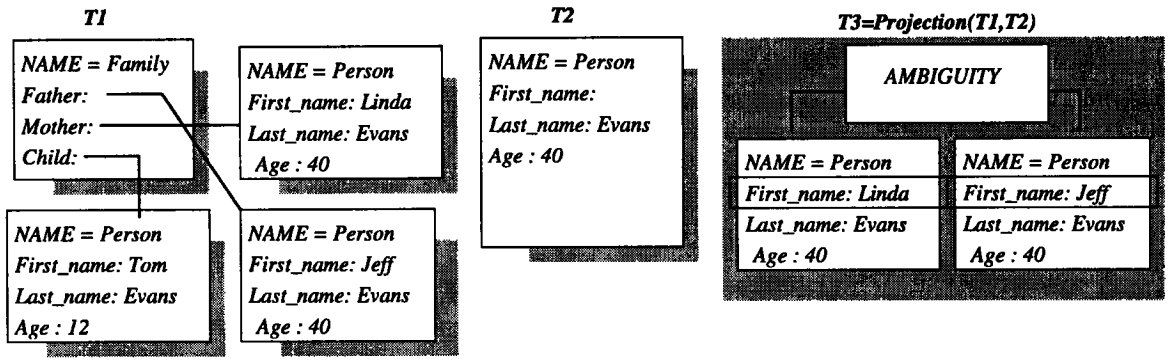

Figure 3: Ambiguity obtained through template projection.

pattern generate causality patterns and repeat step 4.

In our current study we have considered questions only from complexity class 1 and 2.

An Algebra of Dialogue Templates

The main component of the dialogue manager used in INTERLOCUTOR is the task knowledge representation, i.e. the *dialogue templates*. As the dialogue progresses, various instances of the dialogue templates will be generated. A *template instance* is obtained by filling at least one of the slots of a dialogue template. By determining relations among template instances, the dialogue manager determines what queries to pose to the user, what new information is required to resolve ambiguities, etc. This observation determines a very elegant formulation of the dialogue strategies: *no dialogue states or transitions need to be defined, only dialogue motivators.* Dialogue motivators are functions operating on a space of (1) template instances; (2) relations between template instances; and (3) operators on template instances. The template instances can embed other template instances, thus creating a hierarchy. Furthermore, the relations and the operators define a *Template Algebra.*

Template instances are generated by matching linguistic patterns against dialogue utterances. The WordNet-based automatic method of obtaining linguistic patterns that we propose is summarized in the steps of the following acquisition procedure:

1. Build a Semantic Space for every template T
1.1. Retrieve morphological variations for concepts in T
1.2. Select concepts with maximal coverage of relations to concepts from T
1.3. Insert selected concepts in T and goto 1.1
1.4. Derive thematic/contextual information from glosses
2. Determine syntactic contexts for each pair of concepts from T by matching patterns into texts.
3. Generate linguistic patterns based on syntactic contexts
4. Generate semantic constraints for each linguistic pattern by finding the WordNet class of each concept

The application of linguistic patterns generated by this procedure creates template instances. Figure 2 illustrates a dialogue template relevant for the example question of complexity class 2 exemplified in the previous section, as well as a linguistic pattern that fills its slots. The Figure also illustrates two template instances. It may be noticed that both templates and template instances may have slots that are also templates.

Six relations and two operators structure the template instances into a Template Algebra. Each template is defined to have three fields: a *Name*, identifying the discourse template that it instantiates, a *slot-body*, representing all the slot names, and each slot may have a *slot-value*. All slots that are filled with values are collected into a *filled-slot-body*. Sometimes, slot-values are templates themselves. The values of discourse templates may be only other discourse templates. The six relations are:

◇ *Equality*(T_1, T_2) holding when both template instances T_1 and T_2 have the same name and the same filled-slot-bodies.

◇ *Restriction*(T_1, T_2) holding when T_1 and T_2 have the same name but *filled-slot-body*$(T_1) \subset$ *filled-slot-body*(T_2).

◇ *Containment*(T_1, T_2) is true when *Name*$(T_1) \in$ *filled-slot-body*(T_2), thus they do not share the name. Moreover, *filled-slot-body*$(T_1) \subset$ *filled-slot-body*(T_2).

◇ *Generalization*(T_1, T_2) is *Containment*(T_2, T_1).

◇ *Symmetric-Generalization*(T_1, T_2) is true if both *Generalization*(T_2, T_1) and *Generalization*(T_1, T_2) are true.

◇ *Containment-Generalization*(T_1, T_2) holds if there is a sub-part $b \in$ *filled-slot-body*(T_1) such that *Symmetric-Generalization*(b, T_2) is true.

The two operators from the Template Algebra are the *unification*, and the *projection*. The unification generates a new template instance that has slot values resulting from the unification of the argument slots. The projection operator generates a template containing only certain required (projected) slots. The projection operators is extremely important, as it detects ambiguities in the dialogue. For example, Figure 3 illustrates the projection of template T_1 through template T_2, resulting into an ambiguous construct.

AMBIGUITY is a special template, that does not

have any slot-name, but is filled with other templates.

Dialogue Motivators

A dialogue motivator determines what action the dialogue manager should take in conducting its interaction with the user. Initially we shall use the seven motivators implemented in the HMIHPY dialogue manager, reported in (Abella and Gorin 1999). They are *disambiguation, confirmation, error handling, missing information, context switching, continuation* and *Q&A querying.* The *disambiguation* motivators determine when there is ambiguous semantic information, like conflicting locations for the *Taj Mahal.* The *error handling* motivator performs recovery from misrecognition or misunderstanding. *Missing information* determines what information to ask the user in order to satisfy his request. *Context switching* gives the ability to realize that the user has changed his mind or realizes a misunderstanding. The *continuation* motivator determines when it is valid to give the user a choice to ask the Q&A system for additional information. *Q&A querying* decides when the system has acquired enough information to ask the Q&A system a new question.

As an example, we detail the functionality of the *disambiguation* motivator, when called by a template T, the current question template T(q), the last answer template T(a) and the set of templates that represent information that neither the user nor the Q&A system can provide. Such templates are called Templates(IDK), where IDK stands for "I Don't Know". The steps of the motivator are:

Disambiguation Motivator (T,Templates(IDK),T(q),T(a))
1. *If (Restriction(T, Ambiguity)) or*
   (Generalization(T,ERROR))
   Pass T to another motivator;
2. *If (Equality(T(a),IDK))*
   Templates(IDK)=Union(Templates(IDK),T(q))
3. *If (Containment-generalization(T(a),T))*
   Return Projection(T,T(a))
4. *Return T(a)*

First, the motivator checks to see if it applies to the T, otherwise it passes T to another motivator. Otherwise, if the T(a) renders some specific information for T, then a projection is applied, to solve the ambiguity. In any other situation, it is clear that no new information relevant to the fillers that generates ambiguities is available, and the template T(a) is returned unchanged.

Evaluation

In (Walker et al.1997) a new paradigm for measuring the performance of a dialogue system was introduced. This paradigm, called PARADISE (PARAdigm for Dialogue System Evaluation) combines a disparate set of performance measures (i.e. user satisfaction, task success and dialogue cost) into a single performance evaluation function. The PARADISE model considers that the ultimate objective of a dialogue is maximum user satisfaction, which can be achieved by maximizing the task success and minimizing the dialogue costs. In our performance evaluation we have ignored the dialogue cost and measured only the task success by comparing dialogues managed by INTERLOCUTOR with *Wizard-of-Oz* dialogues (e.g. managed by a human expert). For this purpose, we computed Attribute Value Matrixes (AVMs), as defined in (Walker et al.1997), to keep track of the main attributes (e.g. template slots) and their values during dialogues. By comparing AVMs produced by the two different dialogue managers, we generated a *confusion matrix* M. If $P(A)$ is defined as the proportion of times the AVMs have agreed, then $P(A) = \sum_{i=1}^{n} M(i,i)/T$, where T is the sum of frequencies $t_1 + t_2 + ... + t_n$, given that each t_i represents the sum of the frequencies in column i of M.

Similarly, if $P(E)$ is defined as the proportion of times the AVMs agree by chance, it is computed as $P(E) = \sum_{i=1}^{n}(t_i/T)$. The performance is computed with the KAPPA coefficient $\kappa = (P(A) - P(E))/(1 - P(E))$. Our experiments were conducted by three graduate students, their dialogues obtaining κ_1=0.83, κ_2=0.66 and κ_3=0.73.

Conclusion

We have introduced a dialogue shell that enables interactive Q&A. The experimental results indicate that users found answers to their questions faster and obtained more information than with a normal Q&A system. The burden of finding the best strategy for negotiating the meaning of the information need is taken from the user and passed to INTERLOCUTOR. Moreover, the open-domain templates provide insights in the knowledge engineering imposed by dialogue systems.

References

Alicia Abella and Allen L. Gorin. Contruct Algebra: Analytical Dialogue Management. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, pages 191–199, 1999.

James F. Allen. The TRAINS Project: A case study in building a conversational planning agent. *Journal of Experimental and Theoretical AI*, 7:7–48, 1995.

Amanda Stent, John Dowding, Jean M. Gawron, Elizabeth Owen Bratt and Robert Moore. The CommandTalk Spoken Dialogue System. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, pages 183–190, 1999.

Marilyn A. Walker, Diane J., Litman, Candance A. Kamm and Alicia Abella. PARADISE: A framework foe evaluating spoken dialogue agents. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL-97)*, 1997.

Marilyn A. Walker, Jeanne C. Fromer and Shrikanth Narayanan. Learning optimal dialogue strategies: a case study for a spoken dialogue agent for email. In *Proceedings of COLING/ACL-98*, pages 1345–1351, Montreal Canada, 1998.