An Information Repository Model for Advanced Question Answering Systems

Vasco Calais Pedro, Jeongwoo Ko, Eric Nyberg, Teruko Mitamura

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave. Pittsburgh PA 15213
vasco@cs.cmu.edu

Abstract

This paper presents the design and implementation of the information repository which is the central core of the JAVELIN open-domain question answering system. JAVELIN is comprised of several modules that perform a wide variety of question answering (QA) tasks, such as question analysis, document and passage retrieval, answer candidate extraction, answer selection, answer justification, and planning. The architecture is designed to support comparative component-level evaluation, so that different strategies for each module can be integrated and tested in a straightforward way. Each time a module uses a particular piece of information to produce an output, a dependency is created. To support answer justification and introspective learning, the system can use this long-term memory to trace the origin of each answer it produces for a particular question. The JAVELIN Repository implements a complete, consistent relational model for all of the information associated with a question answering scenario.

1. Introduction

Question Answering (QA) systems attempt to provide the user with one or more possible answers to a question posed in natural language, using available corpora and a variety of language processing techniques. For efficient and transparent operation, a QA system must store its input, outputs, and intermediate results in a form that can be reused or inspected by the user as part of answer verification. The data model which is chosen for internal storage must be easily extensible as the algorithms used for QA increase in complexity and new processing steps are added. The interface to the stored data must provide straightforward, seamless integration with a variety of other components in the system. In this paper, we describe the design and implementation of the Repository component in the JAVELIN, a QA system that has been evaluated in the TREC 2002 and 2003 QA Track evaluations (Nyberg et al, 2002; Nyberg et al, 2002).
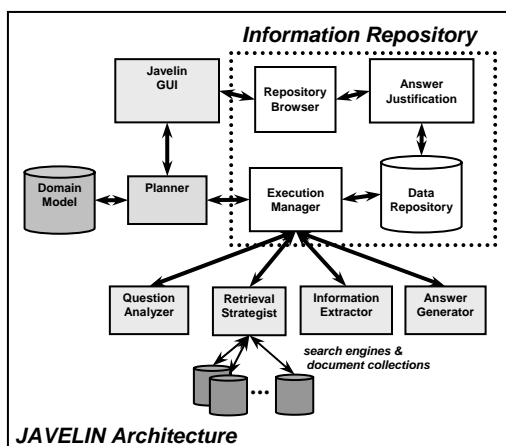

Figure 1 –The JAVELIN architecture and Information Repository

The basic structure of the system is illustrated in Figure 1. The graphical user interface (GUI) accepts a question from the user, and the Planner component interacts with the individual question processing components (Question Analyzer, Retrieval Strategist, Request Filler, Answer Generator) via the Execution Manager component. The Execution Manager provides an abstraction layer for the Data Repository; objects which are stored in static form in a relational database are made available to individual component processors as XML elements.

In Section 2, we summarize the design of the Data Repository. In Section 3, we discuss the implementation. Future work is summarized in Section 4.

2. Repository Design

In this section we discuss the design rationale for the JAVELIN Data Repository, and motivate the use of a relational database for static storage and XML for run-time object manipulation.

2.1. Non-Linear Question Answering

Traditional QA systems followed a pipelined architectural pattern – the question was piped through a succession of modules in a strictly linear fashion, and the final answer(s) produced as the output. Intermediate data structures were not stored in a reusable fashion. More recently, there has been a focus on multi-strategy approaches, where the linear pipeline is replaced with a more flexible control structure that can try several strategies in parallel, or backtrack to an alternate strategy if an answer isn't found (Prager et al., 2003; Harabagiu et al., 2003). JAVELIN takes a non-linear approach to QA; the Planner module decides the next course of action after each processing step is completed. This allows the Planner to adjust when the system reaches a processing state with low estimated utility, without waiting until the final answer set is produced (Nyberg et al., 2003). For example, the Retrieval Strategist may not find a sufficient set of relevant documents for the query that was formulated from the user's question, which might prompt a query refinement dialog. A pipelined approach does not support this type of flexible control. In particular, the intermediate results produced by the modules must be held in a separate data store, and not just passed as arguments to the next module in a pipeline. Furthermore we must store this information in a coherent way that when seen *a posteriori*

represents a consistent chain of reasoning and computation. In addition to the planning steps (strategic decisions) produced by the Planner, the following types of information are produced by the independent JAVELIN modules:

- **Syntactic Analysis**. The Question Analyzer module. This module uses several external tools in order to process open domain text (Nyberg et al., 2003). The resulting syntactic information includes a grammatical feature-structure (f-structure) representation of the question, along with sets of keywords which are later used for document retrieval. The information produced by the Question Analyzer is encapsulated in a Request Object, which is provided as input to other modules in the system.
- **Semantic Analysis**. The Question Analyzer module also produces the expected answer type and a logical representation of the expected answer, based on the f-structure for the question. This information is used by the Information Extractor modules when selecting candidate answers.
- **Document Retrieval**. The Retrieval Strategist uses the keywords produced by the Question Analyzer to create appropriate queries and retrieve relevant documents. The document retrieval information stored in the Repository includes the corpus searched, the ranked list of document IDs returned, and the offsets of the individual keywords in the retrieved documents.
- **Answer Candidates**. The Information Extractors use the retrieved documents and the question analysis to produce a set of candidate answers and supporting passages, as well as a confidence measure for each candidate answer. The Repository design must be flexible enough to capture supporting evidence from multiple extractors for a single answer, as well as multiple sets of answer candidates from a single extractor. Multiple Information Extractor modules may be called by the Planner at different times, yet all the data produced must be related to the same original question.

2.2. Scalability and Flexibility

The scalability of the Repository was a concern for the initial design. We expected that the diversity of information types would increase as new algorithms were researched. We also wanted to support the reuse of information, both for unit-testing individual components, and to tune the system's performance using machine learning approaches. Some of the measures that were taken in order to avoid future bottlenecks included:

- Detailed planning of expected information growth for the first two years.
- Creation of a unique identifier, independent of the object type, for each object in the Repository; this assures the independence between relations and their objects, and supports straightforward insertion of new information types.
- Independent, modular representation of the information produced by the different components, to support non-linear QA scenarios

where different components are called at different times.

2.3. Information Storage for Multilingual QA

The JAVELIN project has explored question answering of English questions from multi-lingual texts (specifically, Chinese and Japanese). This placed an additional requirement on the Repository design, namely that it be able to support multiple data sources in multiple languages. For Chinese and Japanese, we adopted the UTF-8 or Unicode approach in order to store multilingual data. Our implementation chose Microsoft SQL Server 2000 for relational storage and XML DTDs for object representation; both support Unicode characters with little additional effort. The data model was also extended to support additional keywords sets (in multiple languages) produced by the Question Analyzer.

2.4. The Repository as an Independent Language Resource

The information stored in the Repository should be available for introspection and re-use once a question-answering session has ended. The Repository was designed to allow creation of separate web-based applications for browsing the data; this is important both for debugging purposes and for end-user transparency (the user may wish to inspect the intermediate results that led to a particular answer, as part of validating that answer). Since the basic communication with the repository is via XML, it is also straightforward to create new modules that gather and use the existing information as XML objects through communication with the Execution Manager.

3. Repository Implementation

The Repository design was realized in an implementation comprised of a relational database, a set of XML document type definitions (DTDs), and a web-based browsing tool. These elements of the implementation are described below.

3.1. Relational database

The database is the core of the Repository, and is stored in relational format for efficient, secure data access. The relational data model incorporates entities and relations that are general to all question-answering systems, and also includes some entities particular to the JAVELIN system. The Repository stores static information, such as the available set of question types or answer types, as well as dynamic data, such as the passages retrieved for a particular question and the candidate answers extracted from relevant passages. The table in Figure 2 lists some of the most important tables in the database. Figure 3 shows the current sizes of the some of the tables in the Repository, to give an idea of the current scale of the implementation.[1]

---

[1] The current contents of the Repository result from several separate batch runs of the question sets from the TREC QA track test data.

Table Name	Description
Planning	Planning information. Goal and ID of the first planning state chain.
Question	Question string and the date of the question
RequestObject	Generated by the analysis of the question. Contains keyword sets and a logical representation of the question.
QuestionType	The set of possible question types (location, date, etc.). This is a static information type, since the types are part of a fixed (but extensible) hierarchy.
AnswerType	The set of possible answer types. This is a static information type, since the types are part of a fixed (but extensible) hierarchy.
Document	The documents retrieved by the retrieval engine that were considered to be relevant to the question
RequestFill	The candidate answers and passages produced by the information extractors. Contains a passage, a candidate answer, and a confidence measure.
Answer	The answers generated by the Answer Generator, based on a set of Request Fills.
AnswerEvidence	The evidence that supports the answers. This table establishes a relation between answer candidates and answers; there can be multiple pieces of evidence for each answer, or one piece of evidence for multiple answers.
AnswerJustification	A summary of the processing steps taken to achieve an answer

Figure 2 – Description of core Repository tables

Object Type	Total
Questions	6497
Answer Types	17
Documents	52740
Answer Candidates	168908
Answer	80574
Planning Steps	5358

Figure 3 – Current Repository table sizes

3.2. XML elements (DTDs)

Each of the information objects stored in the Repository is also associated with an XML element (DTD), which is used to pass information objects between JAVELIN modules that run as separate, distributed processes in a networked environment. The interface to each module is defined as an XML input / output pair; the modules themselves are freed from the responsibility of directly accessing the Repository. This simplification makes it possible to integrate processing modules implemented in a wide variety of programming environments, without the use of database connectivity middleware (such as JDBC or ODBC).

There are several DTDs, typically one per type of object in the QA process. The XML objects are interpreted by one general module, the Execution Manager, which acts as the communication hub between all the system components. The main advantage of this approach is that it promotes independence of the system's modules, and enables straightforward, centralized refinement of the communications protocol used for system integration. Figure 4 includes an example of the XML object produced by the Question Analyzer.

```
# Question: 290: When was Hurricane Hugo?

# input to the QA server
<QuestionAnalyzer version="0.1" status="OK">
 <Question id="5244">When was Hurricane Hugo?</Question>
</QuestionAnalyzer>

# output from the QA server
<QuestionAnalyzer version="0.1" status="OK">
    <ResourceStats>
        <Time unit="sec">0.01</Time>
    </ResourceStats>

    <Question id="5244"/>

    <RequestObject rank="1" type="initial">
        <QuestionType confidence="0.5">time-expression</QuestionType>
        <AnswerType confidence="1.0">time-point</AnswerType>
        <FStruct>[((orth When)(root when)(pos WRB)(info time_time-point
((orth Hurricane)(root hurricane)(pos NN)(tokens 3)),((orth Hugo)(root
(tokens 5))]</FStruct>
        <Keywords>
            <Keyword type="phrase">
                <Term>Hurricane Hugo</Term>
            </Keyword>
        </Keywords>
    </RequestObject>
</QuestionAnalyzer>

# input to the RS server
<?xml version="1.0"?>
<RetrievalStrategist status="OK" version="0.1">
```

Figure 4 – Question Analyzer output (XML).

3.3. Browsing Tool

The web browsing tool provides a variable degree of detail when browsing through the trace for a specific question/answer pair. The user can view a summary of the question-answering steps, or drill down by expanding specific fields of interest to examine the actual Repository contents. This browsing tool is the result of our initial research efforts to create an effective, visual answer justification. The browser is also an effective debugging tool for system developers. Figure 5 shows the browser's information display in its initial (unexpanded) state.

Figure 5 – An example of the information display available for each question. The user can drill down in the tree of information to reveal additional detail.

4. Future Work

A few issues concerning performance tuning and portability must be addressed in the continued of the development of the JAVELIN Repository:

- The current stored procedures and other access methods used for populating the Repository and accessing stored data during browsing were designed for form and function, rather than speed. Optimization techniques (such as stored procedure tuning and data caching) should be applied before the JAVELIN Repository could be used in a production environment with millions of users and millions of queries per day.

- The Repository is the only component in the current JAVELIN system which runs on Windows and requires a separate software license. In the future, we would like to port the Repository implementation to an open-source database system so that JAVELIN can be freely redistributed without additional cost.

5. Conclusion

In an advanced QA system, each interaction with the user results in a large number of module interactions and the creation of many pieces of related information. This paper described the JAVELIN Repository, which addresses two crucial requirements for advanced QA systems:

- Module traceability and answer validation. The utility of an answer produced by a QA system is directly related to the user's confidence in the sources of information that were searched, and the correctness of the procedures that were used to conduct the search. A system that can show that it searched highly credible sources in a thorough and appropriate manner will inspire more confidence in the user. The Repository provides crucial information for this type of justification, because it stores the individual planning decisions made by the system (what sources to search, what modules to deploy, etc.) as well as the data retrieved. This level of traceability can also support supervised learning - in the future, negative user feedback or correction on a particular decision made by the system can be used to support backtracking, plan revision, etc.

- Consistency and reuse of information. The answers produced by a QA system depend on a) the corpus of documents searched, and b) the algorithms used to conduct the search and analysis of answer candidates. Since both of these may change over time, it is essential that a QA system maintain a traceable history of both information sources and information processing steps. Information produced by the system may be accessed at a later time as part of answering a related question, or it may be determined that previous answers were based on information that is now out of date or invalid.

As we move into more complex question answering and scenario-based question answering, the QA process will require larger amounts of data, bigger document source pools and more complex algorithms. We believe that the design of the JAVELIN Repository will support the required growth and flexibility in the language resources and intermediate results produced by such a system.

Acknowledgements

References

(Harabagiu et. al, 2003). S. Harabagiu, D. Moldovan, C. Clark, M. Bowden, J. Williams and J. Besley, "Answer Mining by Combining Extraction Techniques with Abductive Reasoning, Proceedings of TREC 2003.

(Nyberg et al, 2002) Nyberg, E., T. Mitamura, J. Carbonell, J. Callan, K. Collins-Thompson, K. Czuba, M. Duggan, L. Hiyakumoto, N. Hu, Y. Huang, J. Ko, L. Lita, S. Murtagh, V. Pedro and D. Svoboda, "The JAVELIN Question-Answering System at TREC 2002", Proceedings of the TREC 2002.

(Nyberg et al, 2003) Nyberg, E., T. Mitamura, J. Callan, J. Carbonell, R. Frederking, K. Collins-Thompson, L. Hiyakumoto, Y. Huang, C. Huttenhower, S. Judy, J. Ko, A. Kupse, L. Lita, V. Pedro, D. Svoboda and B. Van Durme "The JAVELIN Question-Answering System at TREC 2003: A Multi-Strategy Approach with Dynamic Planning", Proceedings of TREC 2003.

(Prager et al, 2003). J. Prager, J. Chu-Carroll, K. Czuba, C. Welty, A. Ittycheriah and R. Mahindru, "IBM's PIQUANT in TREC2003", Proceedings of TREC 2003.