A Java implementation of a Question Answering System based on Conditional Knowledge in client-server technology

CRISTINA ZAMFIR (TUDORACHE)
University of Pitesti
Str. Targu din Vale, nr.1, Arges
ROMANIA
cristina.tudorache@star-storage.ro

Abstract: A conditional schema is a graph-based structure which is able to represent conditional knowledge. This structure was introduced in [1]. The inference mechanism corresponding to the conditional schema representations was developed in [2]. A Question Answering System based on conditional knowledge was presented in [3]. In this paper we describe the evolution of that QA System into a client-server application. The conditional schema will be defined and exemplified. The Question Answer System was described in a previous article. In this paper we will describe in a few words the functionality of the QA system and in a more elaborate way the server and the client will be presented step by step in two separate sections. The two sections are oriented specifically towards the task that the described part of the application performs.

Key-Words: - Question Answering System, knowledge base, client-server, natural language

1 Introduction

In prior articles there was presented a practical implementations of a system that interrogates a conditional knowledge base. In [3] there was presented a Question Answering System that can be use in automatic training and system malfunctions diagnostics.

In this paper we will present the Question Answering System evolve into a client server application.

The client/server model provides a convenient way to interconnect programs that are distributed efficiently across different locations. Computer transactions using the client/server model are very common. There are two main entities for every client/server application: one entity is the client and another is the server. Usually the server and the client are found on separate computers.

The client/server programming is based on the idea that the server is a provider of services. Client programs request service from a server by sending it a message. Server programs process client requests by performing the tasks requested by clients. Servers are generally active as they wait for a client request. During these waiting periods servers can perform other tasks. Unlike the client, the server must continually run because clients can request service at any time. Clients on the other hand only need to run when they require service. Many server applications allow for multiple clients to request service.

In this paper we use the client-server technology based on Java mechanisms using sockets and threads. Both the server side and the client side of the application are presented.

The paper is organized as follows: section 2 gives the definition of the Conditional Schema as introduced in [2], the conditional graph and an example of a conditional graph given by an example. Section 3 describes the Question Answering System functions and clarifies how a normal user and an administrator can work with the system. Section 4 gives some general information about the client-server technology and the sections 4.1 and 4.2 explain step by step the actions performed by the server and by the client. Section 5 contains the conclusions and some possible extent of this paper.

2 Conditional Knowledge

The concept of conditional knowledge was introduced in [1]. The inference mechanism of this structure is treated in [2].
A conditional schema is a tuple \( S = (\text{Ob}, \text{Cs}, \text{Er}, A, V, B_{cr}, h, f) \) such that:

- \( \text{Ob} \) is a set of the object names. This set is divided into two subsets \( \text{Ob}_{\text{ind}} \) and \( \text{Ob}_{\text{abstr}} \) such that \( \text{Ob} = \text{Ob}_{\text{ind}} \cup \text{Ob}_{\text{abstr}} \) and \( \text{Ob}_{\text{ind}} \cap \text{Ob}_{\text{abstr}} = \emptyset \);
- \( \text{Cs} \) is a finite set of symbols named conditional symbols;
- \( \text{Er} \) is a finite set of symbols used to designate conditional binary relations over \( \text{Ob} \);
- \( A \) is a set of attribute names for the elements of \( \text{Ob} \);
- \( V \) is a set of values for the elements of \( A \);
- \( B_{cr} \subseteq 2^{((\text{Ob} \times I) \times (\text{Ob} \times I)) \times (\text{Cs} \cup \{T\})} \) is the set of the conditional binary relations;
- \( h : \text{Er} \rightarrow B_{cr} \) is a mapping that assigns a conditional binary relation for every symbol of \( \text{Er} \);
- \( f : \text{Ob}_{\text{ind}} \rightarrow 2^{A \times V} \) is a mapping that assigns initial knowledge to the individual objects of \( \text{Ob}_{\text{ind}} \).

The conditional graph ([3]) generated by \( S \) is the system \( G_S = (X \cup Z, \Gamma_X \cup \Gamma_Z) \) where:

- \( X \subseteq \text{Ob} \times I \) is the set of nodes such that \( x \in X \) if and only if \( \exists r \in \text{Er}, y \in \text{Ob} \times I \) such that \( (x,y) \in_x h(r) \) or \( (y,x) \in_x h(r) \);
- \( \Gamma_X \subseteq X \times \text{Er} \times X \) and \( \left( (n,w_1), r, (m,w_2) \right) \in \Gamma_X \) if and only if \( \left( (n,w_1), (m,w_2) \right) \in_x h(r) \); the elements of \( \Gamma_X \) are named arcs of first category;
- \( Z = \{ f(x) \mid x \in \text{Ob}_{\text{ind}} \} \) and \( \Gamma_Z = \{ (f(x),x) \mid x \in \text{Ob}_{\text{ind}} \} \); the elements of \( \Gamma_Z \) are named arcs of the second category.

Let us consider the following knowledge piece:

```
Jane is 37 years old. She teaches math. Jane is the mother of Dana. Dana has big grades in math if her mother is a math teacher. Jane likes to read SF novels. Jane has a brother named Tom that has a son named Mathew. Tom is 35 years old and he is a school director. If Mathew’s dad is a school director then Mathew has to study hard to get big grades."
```

The graphical representation of this knowledge piece is shown in Fig. 1. The individual nodes are marked by introducing the letter “i” next to the object name. The abstract objects are marked with the letter “a”. The arks are named by the relations they signify.

![Fig. 1 - The graphical representation of the knowledge piece](image)

### 3 The Question Answering System

The QA system is a natural language application that is constructed in Java and interacts with a user through a graphical interface. The application can be used by two types of users: an administrator that configures the system and an end user that consults the application in order to access the desired information.

The architecture of this application has already been presented in [4].

The end user interacts with the application by using the graphical interface shown in Fig. 2. The user has to enter a question in the text field and press the “Ask” button. The question will be transferred to the Analyze Module that will verify the text and will extract the semantics. If the text is
not grammatically correct then the message received from the user will be considered invalid. This information will be passed to the Answer Generation Module that will communicate an appropriate message to the user. If the text received from the user is valid, the Analyze Module will transfer it to the Inference Engine that will perform the inference on the objects. After the inference is performed, the result found by the Inference Module will be transferred to the Answer Generation Module that will transform the answer into a natural language text. The text will be transferred to the graphical interface and the user will be able to see the answer in the text area.

Fig. 2 – The end user graphical interface

The administrator (also called knowledge engineer) can configure the application through a graphical interface that is exemplified in Fig. 3 or directly by editing the xml files.

Fig. 3 – QA System – Administration Graphical Interface

Through the administration graphical interface the knowledge engineer can create a conditional schema, add new components to the conditional schema, modify the existing components, delete the components, visualize the schema, delete the conditional schema, and define, modify and visualize the mapping functions.

All the above can also be done by directly writing the xml files that the application uses to retain all the conditional schema information. The xml files that were presented in [4] are used by the application in order to make it more general and configurable. An example of the xml files used in this version of the application is shown in Fig. 4.

Fig. 4 – An example of the conditional schema xml file

For example, on the knowledge piece given in the “Conditional Knowledge”, the user can ask questions like “Is Jane a math teacher?”. The answer for this question is very simple to find and so the system will answer that “Yes, Jane is a math teacher”.

The end user can ask “Does Dana have big grades in math”. The system will compute and the answer will be “Yes, Dana has big grades in math”.

4 QA Client Server
Because of the fact that the Question Answering system is intended to be used in health consultancy and other complex systems it is only normal for the application to evolve into a client server system.

The QA system was already structured to be used by two types of users that interact differently with the application.

A client server application uses a server and one or more clients. Fig. 5 represents such a case based on network communication.

Fig. 5 – Client-server architecture

In a client-server architecture the server makes a service request and the server fulfills that request. This is very similar with how the application already functions: the user requests an answer and the system interrogates the knowledge base to provide the answer.

In the following sections we will describe the server and the client functions separately.

4.1 Server side application

In this section we describe the actions of the server. The communication between the server and the client is done using sockets. A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

The actions performed by the server are described in several steps.

- Step 1 – Starting the server: The server is started by the administrator. It then checks for an existing knowledge base. If the knowledge base is not found the server throws an error and cannot start connections with the clients. When the knowledge base is found the server moves on to step 2;
- Step 2 – Starting a connection: The server program creates a new ServerSocket object on a specific port (for example 9090). The socket object is considered to be successfully created only if the server binds to the port. This port can be configured by the administrator so that we can assure that the port is always available for the server only. The server now accepts a connection from the client. The server assigns a thread to each client that attempts to connect to it;
- Step 3 – Process the request received from the client: A connectionHandler object is created. This object will process the requests sent and received from the client. The server thread that is assigned to the client will transmit the query to the Analyze module of the QA Application;
- Step 4 – Compute an answer – The QA Application computes an answer to the client request. The answer is transmitted from the “Answer Generation Module” to the server;
- Step 5 – Send an answer to the client – The server sends the client a message that contains the answer to the question that the client requested.
- Step 6 – Terminate the connection;

The administration of the conditional schema and of the knowledge pieces have remained on the server and the graphical interfaces did not change. Additionally, a graphical interface was created for the server. The interface is shown in Fig. 6:
4.2 Client side application

In this section we describe the actions of the client. We “moved” all the end user functions onto a client application.

The actions performed by the client are described in several steps:

- **Step 1** – Initiation of the client application: The client is started by the end user by accessing the end user graphical interface which is installed on the station that the user owns;
- **Step 2** – Establish a connection with the server: the client sends a connection request to the server. The server allocates a socket that the client will use;
- **Step 3** – Communicate with the server: The client sends a message to the server. Part of this message is the question that the end user has asked. The server will compute the answer to the end user inquiry and transmit a message that contains it to the client.
- **Step 4** – Display the message received from the server: The client extracts the answer from the message sent by the server. The message is displayed in the graphical end user interface.
- **Step 5** – Terminate the connection with the server: The connection will be terminated when the end user closes the interface or after 5 minutes of inactivity (the end user does not use the client application for 5 minutes);

The interface of the client application is exactly the one that was used by the end user prior to the evolution of the system.

The user must enter the question in the text field and press the “Ask” button. The client will send the question to the server while in the end user interface the “Ask” button will be inactivated. After receiving the answer to the question the client will display the answer in the text area. To ask another question the user must press the “Reset” button.

An example of the end user interface is shown below in Fig. 7.

![Server Graphical Interface](image)

**Fig. 6 – Server Graphical Interface**

![End User Graphical Interface](image)

**Fig. 7 – Example of the use of the end user graphical interface**

5 Conclusion and future work

In this paper we described in a few words the functions of a QA System. The inference engine and the knowledge base are built taking into account a knowledge representation method named conditional knowledge. The user can interrogate the system only by sentences concerning the declarative facts. In this article the QA System is adapted to a client-server system. We present step by step the client and the server tasks. The server and the client are presented step by step.

We intend to extend our system to obtain a client server dialogue system based on conditional knowledge. In a future work we exemplify the use of such a system in automatic training, to build health systems consultancy and systems for diagnosing the malfunctions of a device.

**References:**


