Virtual Agent Oriented to e-Learning Processes

Celia G. Róspide¹, Cristina Puente¹
¹ Advanced Technical Faculty of Engineering – ICAI, Comillas Pontifical University, Madrid, Spain

Abstract – In this paper we explore the benefits that a virtual agent could provide to an e-Learning platform. The system uses Natural Language techniques as a basis, which makes it capable of elaborating a flexible answer according to the student requirements. The Software procedures to perform these task are also introduced’s.

Keywords: Virtual agent, Flexible answer, e-Learning.

1 Introduction

The arrival of a new educational model, with less theoretical lessons and more work on the student side, has in many cases caused extra work when solving problems via email or through other means.

To alleviate this problem, in this paper we present a virtual agent oriented towards educational environments. The purpose of this is to solve as many questions as possible, emulating the possible answers that a real professor could give. To do so, we have developed an architecture divided into modules in which the original query is analysed in order to understand the requirements of the student to return a suitable answer.

The construction of a chatbot able to emulate a flexible conversation is highly complex. Nowadays, the so-called virtual robot has not been developed in its totality, and the ‘perfect chatbot’ still does not exist. For this reason, there are several enterprises with good reputations in the computer market, like IBM, working on these types of projects to improve their performance [1].

The idea of a virtual agent to answer questions is not new. An intelligent virtual agent is an entity capable of perceiving its environment and processing those perceptions in a rational way [2]. To be considered as AI systems, they have to fulfill the following characteristics [3]:

- Learn new problems and improve the range of solutions.
- Real time adaptation.
- Analysis of a situation in terms of behaviour, evaluating the possibilities of success and failure.
- Evolve through interaction with the environment.
- Manage great amounts of data in order to choose the best solution.

One of the greatest problems in the implementation of a virtual agent is the design of its ‘personality’, as it has to be original and intelligent. However, the behavior of an agent many times is not what is expected, especially when there are many restrictions or when the problem is not defined precisely.

Among the diversity of virtual agents, we can find the group of Intelligent Tutoring Systems (ITS). These type of agents were created in the eighties with the idea of sharing knowledge in an intelligent way to guide the student in their learning process. So an intelligent tutor can be defined as [4] “a software system which employs AI techniques to represent knowledge, and interacts with the students to transfer it” or as [5] “systems to model teaching, learning, communication and knowledge domain of the specialist and the understanding of the student in that domain”.

Recent research is focused on finding an alternative to a human tutor. However, most of the tutoring systems do not acquire the expected level of rationality due to the difficulty involved with modelling human behavior, beyond the application of the most advanced programming techniques.

Taking these premises as a basis, the proposed chatbot is an ITS, which serves as support in the learning process of a student in a given subject. The system is composed of a hardware and software architecture which is explained in the following sections, as well as the interaction among the software modules that make understanding possible and the composition of a flexible answer. Finally, we will conclude with some guidelines for future work as well as some conclusions about the proposed work.

2 Bot Architecture

The platform used to develop this educational bot was a web environment. The hardware used was typical client-server architecture, where the client is a browser like Mozilla or
Explorer, and the server is an Apache with some other additions like a lexical analyzer.

The software is composed of several modules written in different programming languages (like Php, Javascript, C, Lex), which interact among themselves to understand the user’s question and propose a suitable answer. The software model proposed to carry out this task is a Model-View-Controller, where the model interacts with a MySQL database to extract information, the View part is the interface, written in HTML5 plus Javascript, and the Model contains the interaction with the lexical analyzer plus the specific modules written in Php. In a graphic form, figure 1 represents the interaction of all these components:

![Diagram of software and hardware architecture of the bot](image)

**Fig. 1 Software and Hardware architecture of the bot**

As seen in figure 1, the client part is composed of two parts, the interface, and another algorithm to validate the question. The interface is composed of those graphical elements which serve as input and output for information, and has three elements, an image which varies according to the answer, a text area where the dialog between user and the machine is displayed, and an input textbox where the user must write his query.

The algorithm used to validate a question was programmed using AJAX, and is in charge of sending the query to the server and waiting for an answer to be added at the end of the text area.

The web server is divided in three main parts, the query analysis module, the answer algorithm, programmed using the lexical analyzer FLEX\(^1\), and a terminology database built through an ontology (hand-made), with the terms used in each part of the subject and the platform Wordnet Similarities\(^2\) [6].

### 3 Query Analysis Module

The query analysis module is the one in charge of providing a suitable answer, and is composed of other four sub-modules: bad words, welcome, general information and complex answer (in fig. 1 represented as answer algorithm).

---

\(^1\) [http://www.gnu.org/](http://www.gnu.org/)

\(^2\) [wn-similarity.sourceforge.net/](http://wn-similarity.sourceforge.net/).
When a question is posed, the first thing to be done is to dispatch it to the appropriate module. In this way, the first module that evaluates the query is the bad words’ one.

This algorithm detects if there are any bad words within the input received. If so, the user will be warned up to a maximum of three times, after which they will be expelled. To do so, we have implemented a procedure that checks the query words against a database table with a complete list of these words. The result is as seen in figure 2, where the user proposes three questions with some of these words mixed in:

![Fig. 2 Example of bad words output (grey text)](image)

If there are no bad words in the query, the input query will access the welcome module. This algorithm notices if the user is being nice or polite to it, separating the technical query (if one exists) from the welcoming greeting. In this case the answer will be a composition of answers. As in the previous case, the number of times that the user says hello to the bot is controlled, in order to focus the conversation. If the user repeats a greeting message many times, the system will encourage him to ask more specific questions, without being expelled in this case, as seen in the following figure:

![Fig. 3 Example of welcoming output (grey text)](image)

One of the more frequent uses of a chatbot is to answer FAQs. We have included a set (through the database) of questions related to the subject such as the day of the exam, the hour, class, name of the professors and contact information, and many other useful pieces of information.

This third module, is entirely written in PHP, and by matching the query keywords against those stored in the database is able to answer a wide range of questions proposed in a different way. This algorithm also evaluates the question pronouns, like how, what, where, etc., for instance, if the user begins his question with ‘when’, the system will know that he is asking for time. Added to the textual answer, in some cases the system has additional stored information, like schedules, etc. so this information will be displayed along with the answer in a new window:

![Fig. 4 Example of general information output](image)

If the query is more specific and related to the subject’s lessons, then it will be processed by a lexical analyzer, composed of an automaton which will process the input through a set of states.
4 Query Analysis Module

The query analysis algorithm was developed using the morphological analyzer FLEX plus the C programming language. This program is able to detect and locate certain lexical patterns within a text by means of an automaton.

For our specific purpose, this automaton is composed of six states which will be able to recognize:

- Concepts related to the studied subject
- Terms and keywords specifying what the user wants to know about those concepts (like definitions, how it works, and so on).
- Interrogative pronouns.
- Words and expressions commonly used to quit the system or say goodbye.
- Words and expressions used to express gratitude.
- Clauses used to lead the conversation in a friendly way.

The automaton created in FLEX has a [Pattern]-[Action] structure, so if the automaton detects any of the defined patterns, the associated action, written in C code, will be dispatched.

The defined states control the structure of the input query, so in the first place an interrogative pronoun should appear, followed by a word or set of words which will determine what the user wants to know specifically about a concept related with the subject, followed by a concept related with the subject. The query “What is a token?” follows this structure, but not all queries have to, for example “How does a lexical analyzer work?” where the concept related to the subject is not at the end of the query. To take these problems into account, we have created six states which control the possible structures of a query.

In order to ascertain the answer to the proposed question, we have classified the types of questions on the basis of their type and priority:

- Low priority: those not related with the subject’s contents.
- High priority: those containing keywords or relevant patterns.
- Guiding questions: when there is not a clear answer to the proposed question, the system will answer back with another question to check if the prediction was correct. Otherwise it will encourage the student to propose the question in a different way.

This way, if the input is correct, the algorithm will recognize the type of question and a set of keywords, and will look up the related information in the database.

The database algorithm will compare the type of question first, and with the possible results will match the input keywords with those contained in each record. The ‘priority’ of the matches will be taken; so many times a record that has two keywords with high priority matched will be selected instead of another with three keywords with lower priority. To do so, we have used a scoring system. The record that after the matching obtains the highest number of points is selected and its answer displayed. If this record has addition information associated, it will be displayed as well, as in the following example, where the user asks “What is a token?” and the bot gives him the definition along with the slide that contains more information related:

The defined states control the structure of the input query, so in the first place an interrogative pronoun should appear, followed by a word or set of words which will determine what the user wants to know specifically about a concept related with the subject, followed by a concept related with the subject. The query “What is a token?” follows this structure, but not all queries have to, for example “How does a lexical analyzer work?” where the concept related to the subject is not at the end of the query. To take these problems into account, we have created six states which control the possible structures of a query.

In order to ascertain the answer to the proposed question, we have classified the types of questions on the basis of their type and priority:

- Low priority: those not related with the subject’s contents.
- High priority: those containing keywords or relevant patterns.
- Guiding questions: when there is not a clear answer to the proposed question, the system will answer back with another question to check if the prediction was correct. Otherwise it will encourage the student to propose the question in a different way.

This way, if the input is correct, the algorithm will recognize the type of question and a set of keywords, and will look up the related information in the database.

The database algorithm will compare the type of question first, and with the possible results will match the input keywords with those contained in each record. The ‘priority’ of the matches will be taken; so many times a record that has two keywords with high priority matched will be selected instead of another with three keywords with lower priority. To do so, we have used a scoring system. The record that after the matching obtains the highest number of points is selected and its answer displayed. If this record has addition information associated, it will be displayed as well, as in the following example, where the user asks “What is a token?” and the bot gives him the definition along with the slide that contains more information related:

The defined states control the structure of the input query, so in the first place an interrogative pronoun should appear, followed by a word or set of words which will determine what the user wants to know specifically about a concept related with the subject, followed by a concept related with the subject. The query “What is a token?” follows this structure, but not all queries have to, for example “How does a lexical analyzer work?” where the concept related to the subject is not at the end of the query. To take these problems into account, we have created six states which control the possible structures of a query.

In order to ascertain the answer to the proposed question, we have classified the types of questions on the basis of their type and priority:

- Low priority: those not related with the subject’s contents.
- High priority: those containing keywords or relevant patterns.
- Guiding questions: when there is not a clear answer to the proposed question, the system will answer back with another question to check if the prediction was correct. Otherwise it will encourage the student to propose the question in a different way.

This way, if the input is correct, the algorithm will recognize the type of question and a set of keywords, and will look up the related information in the database.

The database algorithm will compare the type of question first, and with the possible results will match the input keywords with those contained in each record. The ‘priority’ of the matches will be taken; so many times a record that has two keywords with high priority matched will be selected instead of another with three keywords with lower priority. To do so, we have used a scoring system. The record that after the matching obtains the highest number of points is selected and its answer displayed. If this record has addition information associated, it will be displayed as well, as in the following example, where the user asks “What is a token?” and the bot gives him the definition along with the slide that contains more information related:

The defined states control the structure of the input query, so in the first place an interrogative pronoun should appear, followed by a word or set of words which will determine what the user wants to know specifically about a concept related with the subject, followed by a concept related with the subject. The query “What is a token?” follows this structure, but not all queries have to, for example “How does a lexical analyzer work?” where the concept related to the subject is not at the end of the query. To take these problems into account, we have created six states which control the possible structures of a query.

In order to ascertain the answer to the proposed question, we have classified the types of questions on the basis of their type and priority:

- Low priority: those not related with the subject’s contents.
- High priority: those containing keywords or relevant patterns.
- Guiding questions: when there is not a clear answer to the proposed question, the system will answer back with another question to check if the prediction was correct. Otherwise it will encourage the student to propose the question in a different way.

This way, if the input is correct, the algorithm will recognize the type of question and a set of keywords, and will look up the related information in the database.

The database algorithm will compare the type of question first, and with the possible results will match the input keywords with those contained in each record. The ‘priority’ of the matches will be taken; so many times a record that has two keywords with high priority matched will be selected instead of another with three keywords with lower priority. To do so, we have used a scoring system. The record that after the matching obtains the highest number of points is selected and its answer displayed. If this record has addition information associated, it will be displayed as well, as in the following example, where the user asks “What is a token?” and the bot gives him the definition along with the slide that contains more information related:

Fig. 5 Example of general information output

Many times the users are not specific with their questions, so the system will try to guess what they mean to say. The prediction algorithm is in charge of this part. The function of this process is to add flexibility and realism to the communication. So we have implemented the following considerations:

- Create the illusion of listening by introducing substrings of the input query in the answer.
- Admit its ignorance about certain issues.
- Control the date (if it is close to the exam date) to offer review exercises, for example, or make comments about how late it is, or say “Good night” when leaving the system.
- Redirect the conversation when the user asks for questions which have little or nothing to do with the subject.
Introduce friendly substring in the answer like “friend”, “colleague”, “dude”, etc.

Calculate the length of the conversation, and if it is too short, encourage him to solve more problems.

Here is an example of conversation using the prediction module:

```
start and press enter...

## I don't have clear what is a token
>> A token is a class of sequence of characters with a collective meaning. You can see some examples and more information on the image displayed.

## a lexical analyzer?
>> What do you want to know about lexical analyzer?

## what is it
>> It is the first step of a compiler and select the lexical components of a program in base to certain rules or patterns. You can see it main functions and more information on the image displayed.
```

Fig. 6 Example of the prediction module

5 Conclusions

This work has introduced a prototype of an intelligent virtual agent able to answer questions related to an academic subject. Using typical client-server architecture and several modules to analyze questions and answers, the system is able to dispatch, in the majority of cases, a suitable answer for the proposed question.

There are some lines that could be improved upon in future work, like proposing exercises to the student based upon their scores, or creating a specific study plan for every student to be controlled through the bot.

6 Acknoledgements

This work has been partially supported by TIN2010-20395 FIDELIO project, MEC-FEDER, Spain and PEIC09-0196-3018 SCAIWEB-2 excellence project, JCCM, Spain.

7 References


