Sādhanā Vol. 41, No. 10, October 2016, pp. 1173–1178
DOI 10.1007/s12046-016-0544-1

Implementation of an inquisitive chatbot for database supported knowledge bases

S RESHMI* and KANNAN BALAKRISHNAN

Department of Computer Applications, Cochin University of Science and Technology, Kochi 682 022, India
e-mail: reshmis@gmail.com; mullayilkannan@gmail.com

MS received 1 April 2015; accepted 25 April 2016

Abstract. Chatbot is a piece of software that responds to natural language input and attempts to hold a conversation in a way that imitates a real person. Some chatbots are used for entertainment purposes, while others for business and commercial purposes. Chatbots are getting a lot of attention from business community right now as they can save costs in customer service centers and can handle multiple clients at a time. Successful implementation of a chatbot calls for correct analysis of user's query by the bot and the formation of the correct response that should be given to the user. In many scenarios the information available from the user's query is inadequate to provide the answer. In such contexts, the chatbot needs to be inquisitive so that it will be more interactive and can mimic a more natural human interaction. This paper reports the implementation of an inquisitive chatbot, which finds the missing data in query and probes the questions to users to collect data that are required to answer the query. Through this implementation, the level of interactivity between the user and the chatbot is improved.

Keywords. Chatbot; intelligent conversional agents; knowledge base; AIML.

1. Introduction

One of the important goals of the researches in the field of human–computer interaction is the design of natural and intuitive interaction modalities. In particular, many efforts have been devoted to the development of systems to interact with the user in a natural language. Computer-based chatbots are becoming popular as an interactive communicative system between human and machines. Chatbot is an artificial entity that is designed to simulate an intelligent conversation with human partners through their natural language and is considered to be one of the classical interfaces for natural language interactions between man and machines. A chatbot, in general, employs a simple text interface for interaction, although some include complex systems such as speech or gesture recognition and text-to-speech features.

Chatbots are becoming increasingly important for scientific, commercial, and entertainment systems. They have a wide range of applications such as virtual assistance, artificial tutoring, e-commerce, and social networking and have the potential to revolutionize the way the human–computer interactions takes place. Currently, chatbots are used by thousands of web users to mediate access to data or knowledge bases and also to carry out generic conversations [1].

Chatting bot can understand what you are saying, analyze it, and give you a suitable response. Some chatbots strive to be indistinguishable from humans, while others try and stand out from humans with super-human knowledge or features. Most chatbots simply look for keywords, phrases, and patterns that have been programmed into their databases, but some use more advanced techniques. As of yet, no chatbot has been able to completely fool humans into believing it is one of them through its knowledge of natural language.

2. Chatbot architecture

Chatbot analyzes the user input and gives a suitable response using natural language processing (NLP) and artificial intelligence. Most of the chatbot systems use some form of NLP by matching the user's input against a knowledge base of words and phrases and select a suitable response based on the input and the context of the conversation. Pattern matching, finite-state-machines, and frame-based models are the main methodologies of conversional agent design [2].

Chatterbots mainly consist of three parts: a knowledge base that encapsulates the intelligence of the system, a chat engine as an interface engine, and an interpreter program [3]. Interpreter program comprises an analyzer and a

*For correspondence

generator for communicating with the user interface. Analyzer reads the input dialog from the human partner and analyzes the syntax and semantic of the sentence. The analyzer acts as a preprocessor to the user input and uses different normalization techniques such as pattern fitting, substitution, and sentence splitting. The chat engine tries to match the preprocessed output of the analyzer and identifies the suitable answer using pattern-matching algorithms with the help of the knowledge base. Knowledge base is the reservoir of an intelligent agent system and it is composed of keywords/phrases and responses associated with each keyword/phrase. Common implementation of knowledge base involves the use of dat files or text files, databases, and XML files. Generator processes the response given by the chatbot engine and generates an appropriate grammatically correct sentence to display. Figure 1 illustrates the typical components of a chatbot and the relation between these components.

3. Existing chatbots

The very first chatbot, named Eliza [4] by Joseph Weizenbaum, simulated a Rogerian psychotherapist. It was inspired by the ideas of Turing [5], who argued that it was possible to build machines capable of acting like humans. The idea was simple and consisted of a pattern-matching algorithm and sentence reconstruction, without in-depth knowledge or processing of natural language. The program proved to be amazingly efficient in sustaining people's attention during the conversation and the success of the original program had influenced the development of many other bots.

Colby from the Stanford Al Lab developed Parry chatbot. Parry is the opposite of Eliza in the sense that it simulates a patient and has been intended as a study of the nature of paranoia and is capable of expressing beliefs, fears, and anxieties [6]. Another implementation called Jabberwacky can learn new responses based on user interactions, rather than being driven from a static database like many other existing chatbots [7]. The general AI of Jabberwacky keeps track of everything people have said to it, and finds the most appropriate thing to say using contextual pattern-matching techniques [8].

Another chatbot worth mentioning is artificial linguistic internet computer entity (ALICE) [9], that has its own markup language called artificial intelligence markup language (AIML), developed by Dr. Richard Wallace, and earned the Loebner Prize in 2000 and 2001. ALICE applies heuristic pattern-matching algorithm to input to obtain a matching pattern in AIML, and this algorithm uses a depth-first search technique with backtracking. The knowledge base of this system is composed of AIML files, which is an extension of the widely used XML format. The corresponding template of the matched pattern in knowledge base is used for the output generation.

Analyzing chatbot development on the basis of NLP, one can identify three generations of these systems [1]. The first-generation chatbots were based on simple techniques of pattern matching such as ELIZA [4]. The second generation includes techniques of Artificial Intelligence and the third generation uses more sophisticated pattern-matching techniques based on markup language. A vast number of third-generation chatbots are based on AIML such as LUCY [10], CHARLIE [11], and Xiao Hui-hui [12]. One of the famous third-generation chatbots is ALICE, which won various awards.

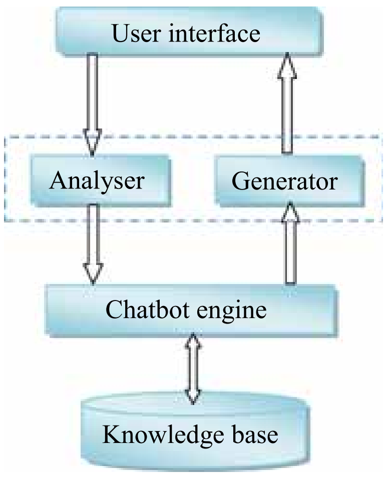


Figure 1. Chatbot architecture.

3.1 *Artificial intelligence markup language*

AIML is an XML dialect with its own specification developed by Richard S Wallace during 1995–2000. This is an XML-compliant dialect for encoding the behavior of bots in a standardized form that can be exchanged between different chatbot interpreters and implementations [13]. AIML is highly recursive, and typically a single input-response pattern will have many alternative pattern matches that resolve recursively to the same ultimate code.

AIML describes a class of data objects named as AIML objects and partially describes the behavior of computer programs that process them. AIML objects are made up of units called topics and categories, which contain parsed or unparsed data. AIML-based agents count on a dialog base of categories formed of units identified by the ⟨category⟩ element.

Each category is composed of an input pattern, by the ⟨pattern⟩ element, associated with one or more output templates identified by the ⟨template⟩ element. The optional ⟨that⟩ tag refers to chatbot's previous reply and through ⟨srai⟩ tag can direct different input patterns to the same exit template. Figure 2 illustrates a basic unit or topic in the AIML format.

```
<aiml version="1.0.1">
<topic name="The topic name"
        <category>
        <pattern>User Input</pattern>
        <that>Last response</that>
        <template>Chatbot answer</template>
        </category>
        <category>
        <pattern>User input * </pattern>
        <template><srai>User Input</srai></template>
        </category>
        ….
</topic>
</aiml>
```

Figure 2. Basic AIML format.

3.2 *Issues and challenges*

Generally, chatbots can respond to a set of predefined queries, which are generally stored in a knowledge base. The ability of the chatbots to respond to the variants of the same query makes it more efficient and attractive, and this ability, to a large extent, is determined by algorithms used for pattern matching and the implementation of the knowledge base. AIML, as discussed before, provides a better platform for such query handling.

At present, the agents can answer only the queries, and cannot probe the user meaningfully for further information collection. This is a very important aspect of a chatbot implementation, mainly for two reasons: one, it could make the conversations more natural and user-friendly; the second reason is attributed to the fact that the user's query may not always contain sufficient information to give an answer. In other words, the question is ambiguous and the bot needs more information to properly give an answer to the query.

4. Proposed system

Sometimes the information available from the original query of the user is inadequate to provide the answer, and in such contexts the intelligent system should be able to identify that some relevant information is missing and should be able to probe the user further to collect the missing information so that the original query can be answered. This paper discusses the implementation of such an intelligent inquisitive chatbot, developed by modifying the ALICE engine, where the knowledge base is supported by a database.

Probably the user's query may contain one or more missing data; in such a situation, the bot needs to come back with one or more question for collecting the missing information. The number of missing information and inquisitive query are directly proportional.

$$D_{\mathrm{n}} \propto Q_{\mathrm{n}}, \tag{1}$$

where D_{n} is the number of missing information that is required to answer the user's query accurately and Q_{n} is the number of questions need to be asked by the chatbot to the user.

Depending on the number of insufficient information for responding to the user's query, the level of complexity for probing the missing information increases. First-level inquisitive query mechanism will return only one inquisitive question to the user. For the second level, the number of missing information is two, and thus, the chatbot is required to pose two questions to the user.

As an illustration of the level of inquisitiveness, consider the case of a college information center. Typically, in a college, the receptionist handles the queries raised by the user and provides the relevant information related to college. Instead of the receptionist, we can use the chatbot application in this real-time scenario for providing information to the users. Such an interactive chat agent is expected to assist the parents or students for providing information regarding admission, academic transportation, and hostel accommodation.

Consider a query asked by the student.

User: *Who is the principal?*
Bot: *Dr. Srinivas ayer*

Here the system can look up in the knowledge base. There is no ambiguity in the question and thus can directly fetch an answer.

However, consider the following query:

User: *Who is the HOD?*

When the system looks up for an answer, then it can find many HODs and some level of ambiguity exists. Here the missing information is the name of the department. Since the user has not specified the department name, the chatbot needs to be inquisitive and should ask the user for the information. Here the minimum number of inquisitive question required is one, and thus this is a case of first-level inquisitive query mechanism for finding the missing data.

Let us consider another query.

User: *Who is our course coordinator?*

The above query misses two pieces of relevant information required for providing an appropriate answer: department and course name. Hence, we can consider this as a second-level query mechanism. Likewise, the chatbot should identify the missing information and collect it from the user through a set of inquisitive queries. In complex scenarios, may be more than two proactive queries need to be asked by the agent to reach the appropriate answers.

5. Methodology

In the existing chatbots, the chat engine identifies the suitable answer to the users' query using pattern-matching algorithms with the help of the knowledge base. ALICE engine is using AIML as a knowledge base that stores a set of predefined queries and its variants. When an information changes, the AIML also is required to change and this will be very difficult. In order to make these hard-coded answers dynamic, we may seek the help of a hybrid knowledge base model, involving AIML and a database.

In this model, when the user asks a query, the answer can come from either of the two possible knowledge bases: AIML or database. More permanent answers are stored in the AIML, while the frequently changing answers are stored in the database. Since the variant data are stored in the database, frequent changes of AIML are avoided. This model can become more attractive when we think of the possibility of integrating the chatbot with the existing databases of CRM or ERP systems, which are already used for office management, and which will always have the updated information.

In order to achieve this, the proposed system is implementing an additional knowledge base engine (KB engine) to the current system and interfaces this with a database for fetching factual data for responding to certain queries. Since the query is first searched in the AIML, we need to have a mechanism in the AIML to instruct the chat engine, to direct the query to the KB engine for searching the database. The proposed modified AIML constructs provides the means for KB engine processing. Knowledge base engine can find a proper answer from the database using a modified AIML response fetched by the chat engine. This construct can also be used to find missing information, which can be obtained from the user after issuing inquisitive queries by the chatbot. Figure 3 shows the architecture with the main elements involved in the proposed system.

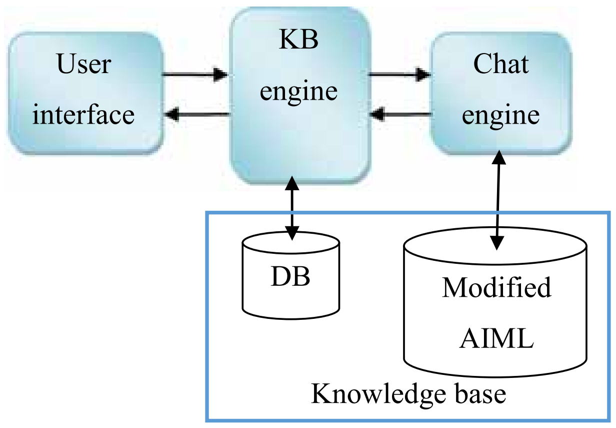


Figure 3. Proposed system architecture.

KB command: F (Command): Main Data: SubData

Figure 4. KB engine command structure.

5.1 *Knowledge base engine (KB engine)*

Knowledge base engine is designed to integrate the database functionality to the AIML and to analyze the missing information on a first level that is required to answer a query and evaluate the responses. The KB engine works on a two-phase evaluation methodology, which involves identifying the missed data field, inquiring the user, and processing the retrieved answer for the formation of right answers expected by the user. First-phase evaluation identifies the missed data and generates the inquisitive query to the user. The second phase processes the user-provided answer for the inquisitive question.

5.2 *Primary phase – missing field identification*

The users' query is intercepted by the KB engine and is passed on to the chat engine for processing. The chat engine, as usual, processes the users' query by parsing it and comparing it against the AIML templates. If the AIML contains the answer directly, it is passed directly to the user, through the KB engine. If the answer should be fetched from the database, the modified AIML template is used, whose format is as shown in figure 4.

The template starts with a token "KB," which indicates that it should be processed by the KB engine. Each command can be mapped with the action that should be executed by the KB engine. F (command) indicates Function to act by KB engine. The template also stores the main and subdata required for processing the action.

This command from this template is provided to the KB engine for evaluation, which will lead to the execution of the appropriate command action.

For finding the missed field, the Main and Subdata section can be made use of. The generalized command response contains the following format:

KB engine command: Function to act by KB engine: Table name which contain the field : (Field required to answering the query).

The identified field can have any value from the set of field value stored in the system. *V* is the set of stored field values represented by

$$V = \{v_1, v_2, \ldots, v_n\}. \tag{2}$$

KB engine compares the user input with the set of available field values.

$$\{f_i\} \cup \{v_1, v_2, \ldots, v_n\} = \phi, \tag{3}$$

where f_i is the user input and $\{v_1, v_2, \ldots, v_n\}$ is the set of available values for the identified field in the system. $\{f_i\}$ should be subset of $\{v_1, v_2, \ldots, v_n\}$. Intersection of set of f_i and set of V is a null set, then the KB engine generates the query with missed data and retort to user interface and set a flag to get the input directly to the KB engine.

5.3 *Secondary phase – provides data from inquired field*

This time the value returned from the user is directly processed by the KB engine instead of chat engine. The KB engine again compares the user input with the set of field values. If any matching field value is found, then concatenate user input with the KB engine command and turn off the flag for getting input value directly to the KB engine. Now the engine gets the missing field value through the inquisitive question to the user, by applying this value it finds the answer to the user's question. For finding the answer, the KB engine again sends the field value and the user's question to the chat engine. This processing requires the saving of the last response from the user; otherwise, this leads to an invalid answer.

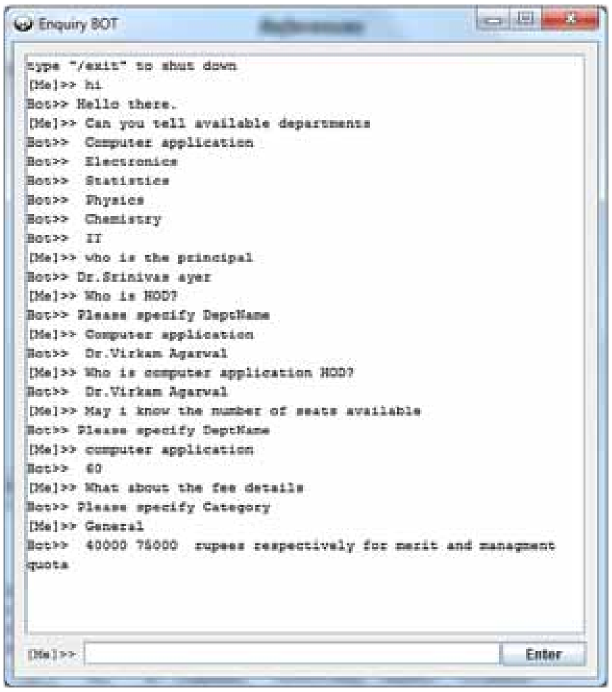
6. Results and discussions

Modified ALICE engine implemented first-level inquisitive ability to the chatbot, which resulted in a more interactivity with the user. After this implementation, we observe the improved interactivity between the user and the agent in the following conversation. The KB engine found the department as the missing data and asked the user to specify the department.

User: *Who is your principal?*
Bot: *Dr. Srinivas ayer*
User: *Who is HOD?*
Bot: *Please specify the department*
User: *Computer application*
Bot: *Dr. Virkam Agarwal*

Consider another conversation, which does not miss the department details in the user query. Here the KB engine is intelligent to identify that the department data are not missed in the user's query and directly provides the answer to the query.

User: Who is computer application HOD?
Bot: Dr. Virkam Agarwal



```
type "/exit" to shut down
[Me]>> hi
Bot>> Hello there.
[Me]>> Can you tell available departments
Bot>>  Computer application
Bot>>  Electronics
Bot>>  Statistics
Bot>>  Physics
Bot>>  Chemistry
Bot>>  IT
[Me]>> who is the principal
Bot>> Dr.Srinivas ayer
[Me]>> Who is HOD?
Bot>> Please specify DeptName
[Me]>> Computer application
Bot>>  Dr.Virkam Agarwal
[Me]>> Who is computer application HOD?
Bot>>  Dr.Virkam Agarwal
[Me]>> May i know the number of seats available
Bot>> Please specify DeptName
[Me]>> computer application
Bot>>  60
[Me]>> What about the fee details
Bot>> Please specify Category
[Me]>> General
Bot>>  40000 75000  rupees respectively for merit and managment
quota
[Me]>>
```

Figure 5. User interface of inquisitive bot.

The user interface of a simple interaction session between a user and the first-level inquisitive chatbot is shown in figure 5.

7. Conclusions

Intelligent conversion agents are becoming popular for scientific, commercial, and entertainment systems. They have a wide range of applications, such as virtual assistance, artificial tutoring, e-commerce, and social networking, and revolutionize the way human–computer interactions take place, while identify the missing data and be inquisitive to the user to collect data that are required to answer the query.

Acknowledgements

The authors gratefully acknowledge the Department of Computer Applications, Cochin University of Science and Technology, for extending all the facilities for carrying out this work.

References

[1] Neves A M M, Barros F A and Hodges C 2006 iAIML: A mechanism to treat intentionality in AIML chatterbots. In: *2006 18th IEEE International Conference on Tools with*

Artificial Intelligence (ICTAI'06), 225–231. IEEE. doi:10.1109/ICTAI.2006.64

[2] Augello Agnese, Giovanni Pilato, Giorgio Vassallo and Salvatore Gaglio 2009 A semantic layer on semi-structured data sources for intuitive chatbots. In: *2009 International Conference on Complex, Intelligent and Software Intensive Systems*, 760–65. IEEE. doi:10.1109/CISIS.2009.165

[3] Hettige B and Asoka S Karunananda 2006 First Sinhala Chatbot in action. In: *Proceedings of the 3rd Annual Sessions of Sri Lanka Association for Artificial Intelligence (SLAAI)*, 4–10

[4] Weizenbaum J 1966 Eliza—A computer program for the study of natural language communication between man and machine. *Commun. ACM* 9(1): 36–45

[5] Turing A M 1950 Computing machinery and intelligence. *Mind, New Ser.* 59(236): 433–460

[6] Güzeldere Güven and Stefano Franchi 1995 Dialogues with colorful personalities of early AI. In: *Constructions of the Mind. Artificial Intelligence and the Humanities. Special Issue of the Stanford Humanities Review*. 4: 161–170. http://web.stanford.edu/group/SHR/4-2/text/dialogues.html

[7] Carpenter Rollo and Jonathan Freeman 2005 *Computing machinery and the individual: The personal Turing test*, vol. 1. http://www.jabberwacky.com/s/PTT100605.pdf

[8] Carpenter Rollo 2004 *Jabberwacky chatbot*. http://chat.jabberwacky.com/j2about

[9] Wallace R S 2001 ALICE. http://www.alicebot.org/about.html

[10] Fei Yi and Stephen Petrina 2013 Using learning analytics to understand the design of an intelligent language tutor – Chatbot lucy. *Int. J. Adv. Comput. Sci. Appl.* 4(11): 124–131. doi:10.14569/IJACSA.2013.041117

[11] Mikic Fernando A, Juan C Burguillo, Martin Llamas, Daniel A Rodriguez and Eduardo Rodriguez 2009 CHARLIE: An AIML-based chatterbot which works as an interface among INES and humans. In: *2009 EAEEIE Annual Conference*, 1–6. IEEE. doi:10.1109/EAEEIE.2009.5335493

[12] Gang Wei Yun, Sun Bo, Sun Ming Chen, Zhao Cui Yi and Ma Pei Zi 2014 Chinese intelligent chat robot based on the AIML language. In: *2014 Sixth International Conference on Intelligent Human–Machine Systems and Cybernetics*. 1: 367–370. IEEE. doi:10.1109/IHMSC.2014.96

[13] Wallace R S and Noel Bush 2005 AIML: Artificial intelligence markup language. http://www.alicebot.org/TR/2005/WD-aiml/