CHATBOTS: CAN THEY SERVE AS NATURAL LANGUAGE INTERFACES TO QA CORPUS?

Bayan Abu Shawar
Information Technology Department
Arab Open University
Amman-Jordan
b_shawar@aou.edu.jo

and Eric Atwell
School of Computing
University of Leeds
Leeds-UK
eric@comp.leeds.ac.uk

ABSTRACT
A chatbot is a program which can chat in natural language, on a topic built into the chatbot's internal knowledge model. Many chatbots exist, with different knowledge-bases programmed by the chatbot builders. We have built a system to convert a website text (corpus) to a chatbot knowledge-base format. In this paper the chatbot is used as a question answer interface, where TRE09 QA track is used to automatically retrain the chatbot knowledge-base. Evaluation shows promising results, 2/3 of generated answers were correct. We aim to see how to improve the algorithm for building the knowledge base before comparing this tool with other natural language interfaces.

KEY WORDS
Chatbot, QA system, human computer interfaces, AIML.

1. Introduction

Human computer interfaces are created to facilitate communication between human and computers in a user friendly way. For instances information retrieval systems such as Google, Yahoo, AskJevees are used to remotely access and search a large information source based on keyword matching, and retrieving documents. However, with the tremendous amount of information available via web pages, what a user really needs is an answer to his request instead of documents or links to these documents. Form here, the idea of question answering systems arises. A question answering (QA) system accepts a user's question in natural language, then retrieves an answer from its knowledge base rather than "full documents or even best-matching passages as most information retrieval systems currently do." [1]. Joshi and Akerkar [2] noted that "a question answering system establishes non-formal communication with the user and answers the queries posed by the user".

However, the main focus of chatbot developers is simulating human-human interaction as realistically as they can. For this reason, Weizenbaum [3] implemented the Eliza chatbot to emulate a psychotherapist. Even though Eliza was based on simple word matching, most of users thought they are talking to a real psychiatrist. After that, Colby developed PARRY [4] to simulate a paranoid patient. "Colby regarded PARRY as a tool to study the nature of paranoia, and considered ELIZA as a potential clinical agent who could, within a time-sharing framework, autonomously handle several hundred patients an hour." These systems were pioneering chatbots, where a chatbot is a conversational software agent which interacts with users using natural language.

Nowadays several chatbots are available online, and are used for different purposes [5] such as: MIA, a German-language advisor on opening a bank account; Sanelma, a guide to talk with in a museum who provides information related to specific pieces of art; Cybelle and AskJevees, web-based search engines, and ALICE for entertainment and casual conversation.

We have techniques for developing new ALICE [6] language models, to chat around a specific topic: the techniques involve machine learning from a training corpus of dialogue transcripts, so the resulting chatbot chats in the style of the training corpus [7], [8], [9], [10], [11]. For example, we have a range of different chatbots trained to chat like London teenagers, Afrikaans-speaking South Africans, loudmouth Irishmen, etc by using text transcriptions of conversations by members of these groups. The training corpus is in effect transformed into a large number of categories or pattern-template pairs. User input is used to search the categories extracted from the training corpus for a nearest match, and the corresponding reply is output.

In this paper, the ALICE chatbot system is adapted to aid as a natural language interface to QA database. In contrast to other QA systems and natural language interfaces that use different natural language processing techniques, ALICE use simple pattern matching techniques based on key word matching. Our ALICE chatbot is retrained with a list of factoid questions obtained from the TREC09 QA track [12].

Section 2 of this paper describes the ALICE/AIML chatbot architecture. Previous work with ALICE is presented in section 3. An overview of other natural language interfaces is in section 4. The way ALICE is retrained with the TREC09 corpus is described in section 5. Section 6 and section 7 discuss the TREC09_Chat results/analysis and conclusions respectively.

2. AIML/ALICE Chatbot Architecture

We have worked with the ALICE open-source chatbot initiative. ALICE is the Artificial Linguistic Internet Computer Entity, originated by Wallace in 1995. A sample of chatting with ALICE is shown in figure 1. ALICE stores knowledge about English conversation patterns in AIML files. AIML, or Artificial Intelligence Mark-up Language, is a derivative of Extensible Mark-up Language (XML). It was developed by the Alicebot free software community during 1995-2000 to enable people to input dialogue pattern knowledge into chatbots based on the ALICE free software technology.

User: Hello, my name is George and I am the human.
ALICE: Hi George.
User: Would you like to play a game?
ALICE: Sure.
User: Ok then. I will email it to you by use of the CD drive.
ALICE: What hobbies do you enjoy?
User: I like cinema.
ALICE: What types of films?
User: All.
ALICE: What movie did you see this year that you liked?
User: I don't have time for movies.
ALICE: I thought you liked cinema.
User: I did, but then you started to say stupid things.

Figure 1. A sample of conversation with ALICE (from Loebner prize contest 2004)

AIML consists of data objects called AIML objects, which are made up of units called topics and categories as shown in figure 2. The topic is an optional top-level element, it has a name attribute and a set of categories related to that topic. Categories are the basic unit of knowledge in AIML. Each category is a rule for matching an input and converting to an output, and consists of a pattern, which represents the user input, and a template, which implies the ALICE robot answer. The AIML pattern is simple, consisting only of words, spaces, and the wildcard symbols _ and *. The words may consist of letters and numerals, but no other characters. Words are separated by a single space, and the wildcard characters function like words. The pattern language is case invariant. The idea of the pattern matching technique is based on finding the best, longest, pattern match.

```
<aiml version=”1.0”>
<topic name=”the topic”>
        <category>
        <pattern>USER INPUT</pattern>
                <that>THAT</that>
        <template>Chatbot        answer</template>
</category>
        ……
</topic> </aiml>
```

Figure 2. The AIML format

2.1 Types of ALICE/AIML categories

There are three types of the AIML categories: atomic categories, default categories, and recursive categories. Atomic categories are those with patterns that do not have wildcard symbols, _ and *, e.g.:

```
        <category>
                <pattern>WHAT IS 2 AND 2</pattern>
                <template> It is 4 </template>
        </category>
```

In the above category, if the user inputs ‘What is 2 and 2’, then ALICE answers ‘it is 4’.

Default categories are those with patterns having wildcard symbols * or _. The wildcard symbols match any input but they differ in their alphabetical order. Assuming the previous input WHAT IS 2 AND 2, if the robot does not find the previous category with an atomic pattern, then it will try to find a category with a default pattern such as:

```
        <category> <pattern>WHAT IS 2 *</pattern>
        <template><random>
                <li>Two.</li>
                <li>Four.</li>
                <li>Six.</li>
        </random></template>
        </category>
```

So ALICE will pick a random answer from the list.

Recursive categories are those with templates having <srai> and <sr> tags, which refer to simply recursive artificial intelligence, and symbolic reduction. Recursive categories have many applications: symbolic reduction that reduces complex grammatical forms to simpler ones; divide and conquer that splits an input into two or more subparts, and combines the responses to each; and dealing with synonyms by mapping different ways of saying the same thing to the same reply as the following example:

```
        <category><pattern>HALO</pattern>
            <template><srai>Hello</srai></template>
        </category>
```

The input is mapped to another form, which has the same meaning.

2.2 ALICE/AIML Pattern Matching Technique

The AIML interpreter tries to match word by word to obtain the longest pattern match, as this is normally the best one. This behaviour can be described in terms of the Graphmaster as shown in figure 3. Graphmaster is a set of files and directories, which has a set of nodes called nodemappers and branches representing the first words of all patterns and wildcard symbols. Assume the user input starts with word X and the root of this tree structure is a folder of the file system that contains all patterns and templates; the pattern matching algorithm uses depth first search techniques:

- If the folder has a subfolder starting with underscore then turn to, "_/", scan through it to match all words suffixed X, if no match then:
- Go back to folder, try to find a subfolder starts with word X, if so turn to "X/", scan for matching the tail of X, if no match then:
- Go back to the folder, try to find a subfolder start with star notation, if so, turn to "*/", try all remaining suffixes of input following "X" to see if one match. If no match was found, change directory back to the parent of this folder, and put "X" back on the head of the input. When a match is found, the process stops, and the template that belongs to that category is processed by the interpreter to construct the output.

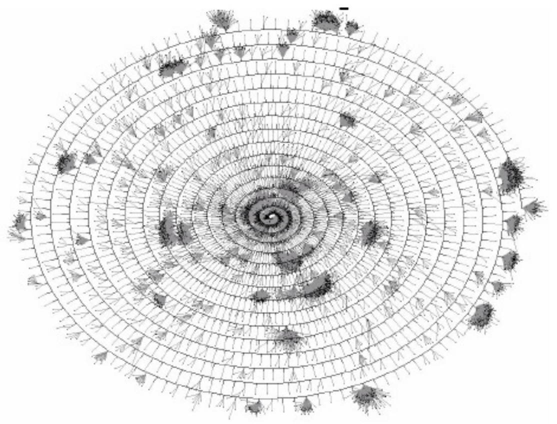

Figure 3. Graphmaster representing ALICE knowledge base

3. Retraining ALICE with TREC09 QA Corpus

The TREC-9 question/answers were used to retrain ALICE. The TREC-09 corpus contains 693 factoid questions with corresponding answers. We trained ALICE with 500 of them and removed other question patterns where the same question was reformatted in different ways. The TREC09 corpus has a standard format where each question is preceded by the keyword "Question" followed a serial number. The answer also is preceded by a special code as shown in figure 4.

Question 205
What is the population of the Bahamas?
AP880510-0226
250,000

Question 206
How far away is the moon?
AP890628-0218
quarter of a million miles

Question 207
What is Francis Scott Key best known for?
AP891213-0037
The Star Spangled Banner

Figure 4. Sample of TREC09 question/answer corpus

This standard format of TREC09 makes the modification process of the Java program easy, as it allowed us to avoid some of the problems in previous work with WWW FAQs such as: using a variety of different tags and annotations to denote questions/answers; and extra annotations encoded indirectly, eg in the file header and footer. Moreover, in the TREC09 corpus no cross-referencing eg URL links are used, so the answer will be just a text without referring users to any web pages to have more information.

The ALICE chatbot engine does not need a linguistic knowledge module, and also in principle is language independent: it can be retrained with Q/A in any natural language. The Java program we developed and used before was adapted to deal with the TREC09 QA corpus. TREC09_Chat [13] was generated; the way TREC09_Chat works is described below:

1. Reading the Q/A from the corpus.
2. Filtering the Q/A by removing unnecessary tags, punctuations and special symbols, and converting the normalized question into capital letters.
3. Building the atomic file which contains the original Q/A
4. Building a frequency list of lexical items in the questions. This list will be used to obtain the first and second most significant words (least frequent words) from each question.
5. Building the default category file. AIML pattern-matching rules, known as "categories", are created. There are two possible types of match: input matches a complete QA question; or input matches 1st or 2nd most significant word in a QA question (least frequent words).

4. Results and Analysis

From the TREC09 corpus, we retrained using 500 QAs. As a result of the program, 500 atomic categories where generated and 6841 default categories. The 500 questions are all matched and correct answers were obtained.

The real aim is to see how TREC09_Chat works if different variants of the questions were entered, which are not there in the ALICE knowledge base. For this purpose, TREC09 variant keys as shown in figure 5 were used to test this ability.

Figure 5 illustrates this. Question number 398 is the original question, and others represent different forms to ask the same question. In total there are 193 different forms related to 54 questions and on average 3 variant forms were tested for each question.

Orig: 398. When is Boxing Day?
815. What is the date of Boxing Day?
816. What date is Boxing Day?
817. Boxing Day is celebrated on what date?

Figure 5. Different patterns of the same question

The original question has the same atomic category, so a match will occur here. Default categories are built based on the idea of the 1st most significant word (1st MSW) and 2nd MSW which are the least frequent words in the question. For example, taking the original question of figure 5, question 398, and then the 1st MSW is BOXING and the 2nd MSW is DAY. The system produces default categories with the following patterns and the same template:

1. <pattern>* BOXING</pattern>
2. <pattern>BOXING *</pattern>
3. <pattern>* BOXING *</pattern>
4. <pattern>WHEN BOXING</pattern>
5. <pattern>WHEN BOXING *</pattern>
6. <pattern>WHEN * BOXING</pattern>
7. <pattern>WHEN * BOXING *</pattern>
8. <pattern>DAY *</pattern>
9. <pattern>*DAY </pattern>
10. <pattern>*DAY *</pattern>
11. <pattern>BOXING*DAY</pattern>
12. <pattern>* BOXING*DAY</pattern>
13. <pattern>WHEN * BOXING*DAY</pattern>

Patterns 1-3 treated the 1st MSW in different positions and the same applied for patterns 8-10 which represents the 2nd MSW. The first lexical item in the question "WHEN" plays the role of question classification. Therefore, patterns 4-7 merge first lexical item with 1st MSW where MSW could appear in the middle of the question or at the end of it. Patterns 11 and 12 concatenate 1st MSW and 2nd MSW with the possibility of being in different positions. Pattern 13 represents the longest match and contains the three most important words: first which represent the question type, and the two most significant words.

It may seem that this approach is illogical; some may say that it will be enough to use patterns 11-13. However, from our previous experience it was better to deal with all situations since we could not guarantee the user input at the end. Assume a question such as: What is your name? in this case, there is only one significant word "name", so pattern 11-13 will not be matched while a match should occur somewhere in patterns 1-10. For this, we enlarge the knowledge base by different possibilities to avoid the usage of any sophisticated natural language processing techniques. We want to demonstrate that this simple technique but with a huge knowledge-base will be sufficient. In all cases, the ALICE pattern matching technique will select the longest match.

TREC09_Chat generated 2/3 correct answers for the variant keys. After a careful analysis for the other 1/3 that were wrong, we found that this is related to the fact that our Java program was initially developed to deal with conversational corpora, and in error cases, chatting may interact with the user in the same topic but not totally specific to the user's question. This will not be acceptable when using a chatbot as a QA interface, because a correct answer is needed, not some chatty remark on the same topic. For example, assume the case that QA corpus has 3 questions about CNN as follows:

Orig: 415. What does CNN stand for?
Orig: 416. When was CNN's first broadcast?
Orig: 417. Who owns CNN?

The generated frequency list from whole corpus includes the following:

Words	Frequency
Broadcast	1
Owns	2
CNN	3
Stand	10
First	19
Does	28
When	40
WAS	79
Who	105
What	232

According to this frequency list, the 1st MSW and 2nd MSW for each question are shown in table 1.

Q_ NO	Question	Answer	1st MSW	2nd MSW
415	What does CNN stand for?	Cable News Network	CNN	Stand
416	When was CNN's first broadcast?	June 1, 1980, we went on the air at 6 p.m., Eastern	Broadcast	CNN
417	Who owns CNN?	Ted Turner	Owns	CNN

Table 1. first and second most significant word in each question

Assume now that different forms of question number 416 as shown in table 2 are entered to TREC09_Chat. We conclude that all these forms give wrong answers, but the answers are still about CNN.

Orig. 416. When was CNN's first broadcast?		
Forms	Frequency in QA corpus	The matched category

CNN began broadcasting in what year?	CNN: 3;
	year 8;
	began: 0;
	broadcasting: 0;
		<category>
		<pattern>CNN*
		</pattern>
		<template>Cable News Network
		</template>
		</category>

1st MSW and 2nd MSW are: CNN and year consequently		

When did CNN go on the air?	air: 1;
	CNN: 3;
	go: 3
		<category>
		<pattern>*
		AIR</pattern>
		<template>July 1 through Labor Day</template>
		</category>

1st MSW and 2nd MSW are: air and CNN consequently.		

Table 2. some of the forms that are entered in TREC09_Chat to represent question number 416

In the first form, MSW are "began" and "broadcasting", but these words are not found in the frequency list, so the system will consider "CNN" as 1st MSW and "year" as 2nd MSW, and consequently a wrong match will be generated. In both cases appearing in table 2, there is no category that combines the two most significant words together in the knowledge base. So the ALICE interpreter will try to find the category that contains one of the significant words, and this is the worst case and usually the answers are not right as shown in table 2 but still in the same topic. To tackle this problem, we need either to enlarge the training corpus so different variants of the lexical items (broadcast, broadcasting, broadcasted) could be found; or to add extra default categories that handle the different variants of MSW.

As future work, we plan to compare TREC09_Chat results with other natural language Q/A interfaces such as START, and ANSWERBUS.

5. Conclusion

We managed to demonstrate that a simple ALICE-style chatbot engine could produce correct results most of the time. We did not need sophisticated natural language analysis or logical inference; a simple (but large) set of pattern-template matching rules is sufficient. The TREC09 QA track corpus was used to retrain ALICE, and new chatbot TREC09_Chat was generated. The standard TREC09 variant questions were tested, and produced correct results in 2/3 of cases. We conclude that in general a chatbot could be used successfully as a natural language interface to a QA corpus, in applications where a large training corpus of Questions and Answers is available, but the exact correct answer is not essential in all cases. For example, many websites have a FAQ page or similar set of questions and answers, which currently can only be accessed via a web-browser, so the user has to trawl through the text to find a question similar to their own; our tools for automating the creation of a chatbot from this source could allow an alternative question-answering mechanism on the website, which though not perfect could be a useful addition.

References

[1] Dumais, S., Banko, M., Brill, E., Lin, J., and Ng., A. 2002. Web question answering: is more always better?. *In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. (SIGIR 2002).* Tempere, Finland, pp. 291-298.

[2] Joshi, M., and Akerkar, R. 2008. Algorithms to improve performance of natural language interface. International Journal of Computer & Applications. Vol. 5, No. 2, pp.52-68.

[3] Weizenbaum, J. (1966). ELIZA-A computer program for the study of natural language communication between man and machine, *Communications of the ACM,* Vol. 10, No. 8, pp36-45.

[4] Colby, K. (1999). Human-computer conversation in a cognitive therapy program. In Wilks, Y. (eds.) *Machine conversations.* Kluwer, Boston/Drdrecht/London. Pp. 9-19.

[5] Abu Shawar, B.; Atwell, E. (2007) Chatbots: Sind Sie wirklich nutzlich? (are they really useful?). LDV-

Forum Journal for Computational Linguistics and Language Technology, vol. 22, pp. 31-50.

[6] ALICE (2002). *A.L.I.C.E AI Foundation*, http://www.Alicebot.org/

[7] Abu Shawar, B., Atwell, E. (2003a). Using dialogue corpora to train a chatbot in: Archer, D, Rayson, P, Wilson, A & McEnery, T (editors) *Proceedings of CL200,* pp.681-690

[8] Abu Shawar., B., Atwell, E. (2003b). Machine learning from dialogue corpora to generate chatbots. *Expert Update*, vol. 6, pp. 25-30.

[9] Abu Shawar, B., Atwell, E. (2003c). Using the corpus of Spoken Afrikaans to generate an Afrikaans chatbot. *Southern African Linguistics and Applied Language Studies.* Vol. 21, pp. 283-294.

[10] Abu Shawar, B., Atwell, E. (2004). An Arabic chatbot giving answers from the Qur'an. In: Bel, B & Marlien, I (editors) *Proceedings of TALN04.* Vol 2, pp. 197-202 ATALA.

[11] Abu Shawar, B. Atwell, E. (2005). Using corpora in machine-learning chatbot systems. *International Journal of Corpus Linguistics* 10:4, pp. 489-516

[12] TREC QA. 2009. [Online]: http://trec.nist.gov/data/qa/t9_qadata.html

[13] TREC09_Chat. [Online]: http://www.pandorabots.com/pandora/talk?botid=81f8f6388e345d52