Towards Improving the Performance of Chat Oriented Dialogue System

Ridong Jiang, Rafael E. Banchs
Institute for Infocomm Research
Singapore 138632
Email: {rjiang, rembanchs}@i2r.a-star.edu.sg

Abstract—This paper is concerned with how to improve the overall performance of chat-oriented dialogue system. This research is motivated by the fact that majority of current chat engines are based on pattern matching. The knowledge base of this type of systems is predefined pattern-answer pairs such as the categories defined in Artificial Intelligent Markup Language (AIML). The inherent disadvantage of this kind of chat engines is that the interaction is carried out without any syntactic, semantic and contextual information. We propose a chat engine which is capable of dynamic knowledge acquisition and inference for a higher level of conversation intelligence. The dialogue engine leverages on natural language processing tasks such as syntactic and semantic parsing, named entity recognition, dialogue act detection, polarity analysis, etc., as well as dialogue history and heuristic rules for analysis and inference to achieve better understanding and intelligence.

Keywords-chat oriented dialogue; knowledge extraction

I. INTRODUCTION

With the advancement of natural language processing and artificial intelligence in recent years, more and more practical spoken dialogue applications are emerging in different do-mains. One of the typical applications is intelligent virtual assistant such as Apple’s Siri, Google Now, Microsoft’s Cortana, and Amazon’s Alexa which makes our daily life much easier. There is an emerging trend of message bots such as Facebook Messenger, WeChat, Line, Telegram, Slack, Kik, etc. These messaging platforms allow users to build customized messaging bots to provide automated services through their messaging interface. The bots can be any kind of services which need conversation interface such as customer support, question answering, online help system, automated tutoring, conversational commerce, etc.

Most of the chatbots nowadays are largely based on the original chatbot ELIZA by Joseph Weizenbaum [1]. The principle of this kind of chatbots is pattern matching which will identify keywords or phrases in the input utterance. These keywords or phrases are associated with rules with predefined responses which are saved in a script file or knowledge file. A typical example of this type of chatbot is ALICE (Artificial Linguistic Internet Computer Entity). ALICE represents its knowledge using AIML [2, 3]. AIML allows the use of categories and topics. User can define the pattern for matching and template for system response. ALICE has won three times (2000, 2001, 2004) the annual Loebner Prize \(^1\) [4].

Mitsuku chatbot created by Steve Worswick\(^2\) is also based on AIML and won the Loebner Prize 2013 and 2016. Chatscript\(^3\) is another chatbot engine used to create various powerful chatbots such as Suzette, Rosette and Rose. Suzette, Rosette and Rose won the Loebner Prize 2010, 2011 and 2014 & 2015 respectively\(^4\). However, the challenge of this kind of systems is to prepare the hand-crafted matching rules. Tremendous effort is also needed for the maintenance of the pattern-based knowledge base. Furthermore, the dialogue management mechanism is too rigid to suit for minor changes or paraphrases of a given utterance conveying the same meaning. Following example in Fig. 1 shows the complexity for preparing the rules of matching “I love you”: \([5]\):

\[
\text{Fig. 1. Sample AIML rules for matching “I love you”}
\]

Four rules have been crafted in order to match all the possibilities of words which may appear at the two sides of “I love you”. However, it will fail when you say “I really love you”. Then one more rule is needed in order to cover the case:

\[
\text{Fig. 1. Sample AIML rules for matching “I love you”}
\]

Problem will arise when the utterance is “I never love you”. It matches the above pattern but it means totally different. That is the true drawback of pattern matching.

Another widely used technique for retrieval based chatbot is similarity calculation \([6]\). Question and answer pairs have been built up in advance as knowledge base for this type of chatbots. Finding the right answer is a process of calculating the similarity between the user’s utterance and the questions in the knowledge base. The results will be ranked and compared with a threshold. Answers will be those question answer pairs whose scores are bigger than the threshold. Compared with pattern matching method, there are not any costly handcrafted rules. However, the order of the terms is lost because of the use of bag of words. Fig. 2 shows the examples which will confuse this type of chat engines.

---

a) “Tom beat Mark”
“Mark beat Tom” \} \quad \text{Same score because of the same bag of words}

b) “Peter is tall”
“Peter is not tall” \} \quad \text{Same score because “not” is a stop word}

Fig. 2. Sample sentences with same similarity score

In Fig. 2 a), both sentences have the same score but with opposite meaning. In the case of b) “not” will be removed in score calculation. That is because “not” is generally considered as a stop word.

For more challenging question such as: “Andy is shorter than Mark and Tom is shorter than Andy. Who is the tallest?”, it is not surprising that the champion bot failed to answer this question in the Loebner Prize Competition 2014\(^5\). To tackle this problem, more advanced natural language processing and inference tools must be used.

We propose a chat engine which is capable of dynamic knowledge acquisition and inference for higher conversation intelligence. The dialogue engine leverages on the results of natural language processing tasks such as syntactic parsing, named entity recognition, dialogue act detection, polarity analysis, etc., as well as dialogue history and heuristic rules for analysis and inference to achieve better understanding and intelligence.

II. APPROACH

The system is composed of the Apollo dialogue engine, a number of plug-in modules and a supporting database, knowledge base and scripts as shown in Fig. 3 [7]. Each module will carry out a specific function towards the final response. For instance, the Parser module will parse the user’s input to produce constituent tree and dependent tree. The parsing results can be passed to the downstream modules such as Knowledge Extraction module through the dialogue manager. The dynamic knowledge extraction results can be fed into the inference engine to generate new knowledge for answering questions. The functional flow of the chat engine is shown in Fig. 4.

A. Parsing and Facts Extraction

Parsing is an important step for better understanding the user’s input. We use Stanford CoreNLP natural language processing toolkit to generate the constituency-based parse tree and the dependency-based parse tree [8]. Together with the information from CoreNLP toolkit such as the POS tags, the coreference resolution, and named entity recognition, they provide very rich and useful information for the downstream analysis modules. For instance, triples which are made up of subject, predicate and object can be extracted from the parsing results. The new extracted triples will be added into the knowledge base which is represented as triple knowledge graph.

Given the following user input:

“John often helps his mother”

The corresponding constituency and dependency trees are shown in Fig. 5. The constituency tree shows the complete decomposition of the sentence down to its most detailed elements with terminal symbols and lexical entries (5a). Dependency tree shows the dependency relations between words and phrases (5b). From the parse trees we can analyze the find out different relations between words or phrases such as the triples: (subject, predicate, object).

In the example of Fig. 5, we can have the following assertion after analyzing the relations. The methods may be different for different systems, though.

help (John, mother)

Internally, the system will keep all information about every word, this is to facilitate future further analysis and answer matching. The representation of the words in the above triples is as follows:

Predicate: Word(word='helps', tag='VBZ', seq=3, lemma='help')
Subject: Word(word='John', tag='NNP', seq=1, lemma='John')
Object: Word(word='mother', tag='NN', seq=5, lemma='mother')

The relation of the triple is shown in Fig. 6. The words “John” and “mother” are graph nodes. The relationship between the nodes is the word “help” which is the edge between the two nodes. A unique identification will be assigned to every triple relation in the knowledge graph. All other information about the words and the unique identification are attached to the nodes or edge as attributes.

With the above triple knowledge graph extraction and representation, following questions could be answered without any handcrafted rules:

Who often helps his mother?
Whom does John always help?
What does John always do for his mother?

---

\(^{5}\) [http://aisb.org.uk/events/loebner-prize](http://aisb.org.uk/events/loebner-prize)
B. Question Type Detection

Question analysis will provide additional clues for the chatbot to find the right answer [9]. For instance, "when" question is mostly asking for the time and "where" question is asking for the location.

For every user input, we use a hybrid method which combines pattern rules and constituency-based parse tree to determine the sentence type.

C. Dialogue Acts Identification

Dialogue act is widely used to describe the user’s intention. That is the illocutionary force of an utterance [10]. For instance, whether the user is greeting, requesting something or it is just a statement. Dialogue act itself is a complex topic for both dialogue act taxonomy and classification [11]. Here we are only concerned about how to use this information to help the system get the correct answer. For instance, if the dialogue act is REQUEST and the question type is WHAT, then the system will be quite clear that the user is inquiring about information. Currently the sets include GREET, STATEMENT, REQUEST, PROPOSE, ACCEPT, PROVIDE, ACKNOWLEDGE etc. Every act can include more specific dialogue attributes such as food, location, etc.

D. Goal Detection

Dialogue acts and topic segmentation provide very useful information for higher-level conversation analysis [11, 12]. The goal of a conversion can be deduced from dialogue act and its attributes. For instance, if the dialogue act is “REQUEST” and its attribute is “HOTEL”, then we know the goal of the user is “looking for a hotel”.

Keeping track of the user’s goal during a conversation is helpful for the system to understand the context. We assume that the user always focuses on the same goal in different dialogue turns unless it is obviously changed. With the correct goal detection, the successful rate of a conversion will improve and the interaction will be more engaging.

E. Inference

Dynamic knowledge extraction and analysis on user’s utterances as described in previous subsections are helpful for the chatbot to discover user’s overall intent as well as to understand specific requirements. However, for more challenging questions such as the question described in section I (Loebner Prize Competition 2014):

Andy is shorter than Mark and Tom is shorter than Andy. Who is the tallest?

Dynamic knowledge extraction is not enough because the answer is not explicitly included in the extracted facts. To answer this question, logic deduction must be conducted so that all possible new assertions can be created. With the new assertions added into its knowledge graph, the chat engine will be able to find the right answer. To enhance our chat engine with the capability of inference, we integrated CLIPS 6 into our system as a plug-in for the inference based on the provided rules and existing triple knowledge. Interfaces were created for firing CLIPS rules in XML and python script and new assertions can be saved into the graph knowledge based.

Reasoning capability makes the system cleverer than pure script-based system. It takes the advantage that the system is capable of acquiring factual knowledge from the utterance dynamically. The CLIPS inference engine supports rules (heuristic knowledge), functions and object oriented programming (procedural knowledge). Forward reasoning is triggered when new facts meet the conditions of heuristic rules [13].

In the above example, the system should be able to extract following facts using the methods depicted above:

Shorter (Andy, Mark)
Shorter (Tom, Andy)

Assuming we have the following rule: “if x is shorter than y, and y is shorter than z, then x is shorter than z”, which can be represented in CLIPS as:

(defrule ShorterTree **
  (shorter ?x ?y)
  (shorter ?y ?z)
  (not (shorter ?x ?z))
=> (ASSERT (format nil "(shorter "\%s" \"%s\")" ?x ?z)))

Then following new assertion will be deduced:
Shorter (Tom, Mark)

To get the answer for above example, we need more rules as the following:

If X is taller than Y, then Y is shorter than X
If X is shorter than Y, then Y is taller than X
If X is taller than Y and X is taller than Z, then X is the tallest
If X is shorter than Y and X is shorter than Z, then X is the shortest

III. PROTOTYPE AND CASE STUDY

A proof of concept prototype has been developed. Most of the plug-ins are developed in C++ and reused for

6 http://clipsrules.sourceforge.net/
this chat engine. The “Parser” and “Knowledge Extraction” plug-ins were developed as python plug-ins for the better access of natural language processing resources.

A case study with both knowledge extraction and inference is shown in Fig. 7.

![Fig. 7. Chat engine interface with inference output](image)

In this case study, the chat engine extracted knowledge from the following user inputs:

a) My hobby is football, do you have any hobby?

b) Sam and Mike had a meeting two days ago. Who just had the meeting?

c) Andy is taller than Mark, Tom is shorter than Mark, Jack Tan is taller than Andy. Who is the tallest?

In the case of a), the fact was extracted and retrieved in the following conversation. That shows that the engine is able to make use of the dialogue history data.

In sentence b), the fact was extracted and immediately used in the current interaction. In sentence c), the answer needs the facts extraction and inference with the rules as we stated in section II. With the related four rules, the engine deduced eleven new facts as shown in Fig. 7 and found the right answer. It even can process more complex cases of the similar type which may be enough to confuse a human for a while, for example:

A is taller than B, E is shorter than D, B is taller than C, F is shorter than E, C is taller than D, who is the tallest?

The system will immediately show that it learned five facts and inferred twenty seven new facts which include the answer: “A is the tallest, F is the shortest”.

IV. CONCLUSION AND FUTURE WORK

We propose an intelligent chat engine which leverages on the state-of-the-art natural language understanding and knowledge engineering for more accurate and intelligent response. Different from the traditional and most popular pattern-matching chat engines such as ELIZA and ALICE whose intelligence is pre-scripted, the proposed chat engine is based on parsing, dialogue act detection, polarity analysis, and goal identification. User’s utterances are analyzed and transformed into fine-grained factual knowledge and added into graph database. Inference will take place if the new acquired facts trigger any rules in the rule base. Case study showed that the proposed chat engine is able to “learn” from user’s utterance and “reason” based on predefined rules to show its intelligence for which the majority of existing chatbots are lacking.

To scale up the proposed chat engine with broader intelligence, more commonsense knowledge is needed. In the future, we will work on enhancing the rule base and making use of existing commonsense knowledge base such as Cyc and ConceptNet.

REFERENCES


[7] Ridong Jiang, Yeow Kee Tan, Dilip Kumar Limbu, Tran Anh Tung and Haizhou Li, A configurable dialogue platform for ASORO Robots, APSIPA ASC, Xi’an, China, 2011


