Implementation of Chatbot for ITSM Application using IBM Watson

Neha Atul Godse
Department of Computer Engineering,
Pune Institute of Computer Technology
Pune, India
neha24godse@gmail.com

Shaunak Deodhar
Department of Computer Engineering,
Pune Institute of Computer Technology
Pune, India
shaunakdeodhar996@gmail.com

Shubhangi Raut
Department of Computer Engineering,
Pune Institute of Computer Technology
Pune, India
rshubhangi52@gmail.com

Pranjali Jagdale
Department of Computer Engineering,
Pune Institute of Computer Technology
Pune, India
pranjali.jagdale@gmail.com

***Abstract*—In the current scenario, the end user of Information Technology Service Management (ITSM) application in software companies has to keep on searching the solution for problems he is facing or else finally generate a ticket since he cannot collaborate with the system by asking questions and getting relevant answers. As a solution to this, we aim to design a chatbot that will be specifically tailored for software firm employees. The chatbot can process input using Natural Language Processing (NLP) and can generate a relevant response which will help the end user to solve his query. The chatbot makes decisions itself to answer user's query with the help of the IBM Watson Conversation APIs. It will also remember the context of the conversation and perform tasks such as creation of ticket on behalf of the user.**

***Keywords*— Chatbot, ITSM, IBM Watson, NLP**

I. Introduction

A chatbot is a software program, which uses Artificial Intelligence to interact with humans. The idea is to simulate human conversations wherein one end is user and another end is a machine. These bots have found applications in various domains such as E-commerce services, medical assistance, recommender systems and educational purposes. They can be integrated into existing application platforms such as Skype, Slack, etc.

ITSM comprises of all the managerial aspects of IT businesses. Nowadays, every software firm has a dedicated team of IT experts which oversee the issues faced by other employees of the company.

In the traditional framework, the end user generates a ticket for any cause, issue or query. This query gets assigned in a queue to some employee at ITSM department. The end user needs to wait until his query gets assigned to ITSM employee. It might take several days because of which he may not be able to proceed with his work. Once the ticket gets assigned, the ITSM employee communicates with the end user employee and solves the issue after which the ticket generated gets dismissed. Refer Fig. 1 for traditional workflow.

In the proposed system, we model a chatbot which can be integrated with any ITSM application. The end user need not generate a ticket every time he faces some issue. Instead, he will be able to have a two-way textual conversation with the chatbot. The input to the chatbot will be in the human spoken language which is processed using Natural Language Processing. NLP is a field of AI that consists of computer understanding and generation of human language.

The IBM Watson Conversation service combines different technologies such as machine learning, natural language processing, and integrated dialog tools to create conversation flows between applications and users. Once the input is given to the chatbot by the user through the chat application, it will be sent to a chatbot plugin through a REST API call. This chatbot plugin then communicates with the IBM Watson Conversation for the respective input and receives the matched output. Any actions only if required are performed in the chatbot plugin and then the output is sent back to the chat application. The chatbot also remembers the context of the conversation as well as the user and can perform required actions accordingly. The end user can carry forward the chat in this way until his query gets solved. In such a way, the end user can receive a faster response as compared with the traditional framework of ITSM application. If the end user is satisfied with the chatbot, the frequency of tickets generated gets reduced and hence the manpower required at ITSM department will be reduced. Sometimes it may happen that the end user remains unsatisfied with the response given by the chatbot. In such case, the end user can ask the chatbot to generate a ticket on behalf of him. In this paper, we describe related work including few implemented chatbots in fields other than ITSM. We then describe our proposed system architecture followed by the implementation of chatbot for ITSM.

II. Related Works

Niranjan et al [1] have proposed an interactive conversational system for students, using the Naïve Bayesian concept. This model acts as a virtual teacher which can answer the query of the student in an interactive way using the chat agent that is used. Overall system design includes a lexical parser to extract keywords, Bayesian theory for categorization of knowledge, Probability theory for responses.

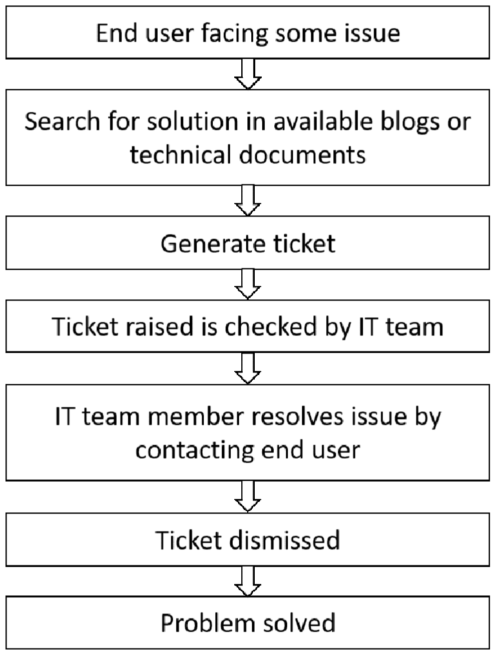

End user facing some issue
Search for solution in available blogs or technical documents
Generate ticket
Ticket raised is checked by IT team
IT team member resolves issue by contacting end user
Ticket dismissed
Problem solved

Fig. 1.Traditional Workflow

Thomas N T [2] has proposed a solution for failing cases of Artificial Intelligence Markup Language (AIML) by adding a Latent Semantic analysis (LSA) block in his design. If AIML fails to answer, the question is transferred to LSA block, where it goes through tokenization and stemming. Virtual agents based on AIML have developed domain-specific bots for the user such as Humorist Chatbot System [4], Dorothy Network Management chatbot, Medical assistance virtual agent [5], Tutorbots, Web based Voice bots [6].

A model was presented by Supratip Ghose et al [3] in which they have investigated ALICE chatbot system to serve a domain-specific service, as aUFAQbot. The UFAQbot could act as an advisor to undergraduates who tend to have doubts regarding Universities and academic advice. The main aim of this system is to reduce the efforts of the user to browse through several web pages to retrieve the necessary information. The ALICE system (Artificial Linguistic Internet Chat Entity) uses an XML dialect called AIML [2] [3]. AIML can handle general questions and greetings but fails in the case of unexpected questions.

By using Natural language processing techniques and machine learning algorithms, there has been an attempt to simulate a human-like conversation with chat agents which can learn from existing conversations. This type comes in intelligent models or generative models for a chatbot.

Deep learning has shown promising results in generative models of chatbot than any other handcrafted chatbots. Chatbots that are developed using deep learning, mostly use a certain variant of sequence to sequence (Seq2Seq) model. Seq2Seq model for chatbot approach was proposed by Oriol et al [7] for conversational modeling. The advantage of this model is that it uses an end-to-end approach that is developing a single model instead of different models for each subtask of chatbot development. Seq2Seq model consists of two types of RNNs namely an Encoder and a Decoder. Given a sequence of inputs X one at a time to encoder, it converts the input to fixed size vector c. Then, the decoder computes the probability of output sequence Y with c as input. Let $X = (x_1, x_2, \dots, x_T)$ and $Y = (y_1, y_2, \dots, y_{T'})$. Seq2Seq model maximizes the generation probability of Y conditioned on X i.e. $p(y_1, \dots, y_{T'}|x_1, \dots, x_T)$. Hence the objective function of Seq2Seq can be given as

$$p(y_1, \dots, y_{T'}|x_1, \dots, x_T) = \prod_{t=1}^{T'} p(y_t|c, y_{1,} \dots, y_{t-1})$$

Vanilla RNNs are incapable of handling 'long term dependencies'. Hence special kind of RNN known as Long Short Term Memory (LSTM) are used. Repeating modules in LSTM chain has a continuously running line called as cell state denoted as C_t. Structures called as gates(sigmoid neural net layer) are used to regulate the flow of information through the cell state. Steps in LSTM include:

1. Forget gate layer to discard unnecessary information. It return value between 0 and 1.
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
2. This step decides new information that is to be added in the cell state. There are two parts in this:
   a. Values are updated using "input gate layer"
   $$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
   b. $tanh$layer creates a vector of new values to add to the state.
   $$\hat{C}_t = \tanh(W_i \cdot [h_{t-1}, x_t] + b_C)$$
3. Update old cell state C_{t-1} into new C_t. Multiply old state (C_{t-1}) with f_t to discard the things that need to be discarded and add $i_t * \hat{C}_t$.
$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t$$
4. Output is the filtered version of cell state. The cell state output is dependent on the result produced by the sigmoid layer.
$$y_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_0)$$
$tanh$function is carried out on the cell state and multiplied with the output of sigmoid layer.
$$h_t = y_t * \tanh(C_t)$$

The model proposed by Iulian V. Serban et al [9] extends the hierarchical recurrent encoder-decoder neural network model and improvises it by bootstrapping the learning from word embedding. Further improvements include topic information incorporation into response generation explained by Chen Xing et al [8].

Some notable developments so far are ELIZA (1966), PARRY (1972) [1], Saya - Japanese chatbot used as a receptionist in a hotel, A.L.I.C.E., Jabberwacky and D.U.D.E (AgenceNationale de la Recherche and CNRS 2006). Not only text-based agents but also voice bots are finding applications in several domains, such as Siri, Google assistant.

III. Proposed System

The workflow of the proposed system is given in Fig. 2.

The system receives input from the end user in the form of natural language. This input is bound in JSON object and passed to the chatbot plugin through a REST API Call. *REST*(Representational State Transfer) refers to a style of web architecture that governs the behaviour of clients and servers through HTTP protocols.

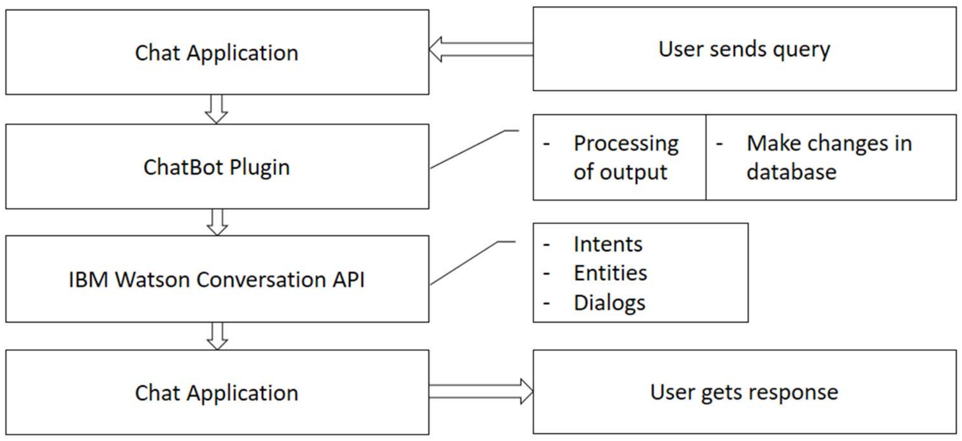


Fig. 2. Proposed System Workflow

The JSON object is unbound in the chatbot plugin and further the input message is sent to the IBM Watson conversation API. The output is returned to the chatbot plugin. Depending upon the output received some operations may take place on the database and the output returned. This processed output is sent back to the chat application. If the user is not satisfied with the reply form the chatbot, it may ask the chatbot to generate a ticket on his behalf. Once the ticket is generated it saves the chat history of the user with the chatbot so that it will be helpful in understanding the context of the query to the ITSM employee to whom the ticket will be assigned. The generated ticket ID is returned to the user so that he may edit, add images or screenshots to the ticket and also check its status.

The different modules in the application are described as follows:

1. Chat application: It accepts the input from the end user in the form of natural language. The input is bound in a JSON object and is sent to the chatbot plugin using REST API Call. This API can be used over nearly any protocol. It also receives the output from the chatbot plugin and displays it to the user.
2. Chatbot plugin: It receives the input from the chat application and forwards it to the IBM Watson Conversation API. Once the output is received it is responsible for making necessary changes into the database server and also process the output if required.
3. IBM Watson Conversation API: It consists of intents, entities and dialogs.
   a) Intent: It represents the purpose of a user's input.
   b) Entities: It represents an object that is relevant to your intents and that provides a specific context for an intent.
   c) Dialog: It is a branching conversation flow that takes place when the API recognizes the defined intents and entities.

   Once the input is received, its intent is identified based on maximum probability. The output is mapped on the basis of entities and the dialog flow. This output is then sent back to the chatbot plugin.

IV. Implementation

A chatbot for ITSM in software companies has been implemented based on the proposed model discussed above. As the chatbot is tailored to be used in software companies, we have considered some sample data such as a virtual office structure, a database of employee details and certain other databases to store the state or context of the system. For sample virtual office layout refer Fig. 3. The designed virtual office demonstrates the location of employee according to designation, the locations of devices that are used nearby such as ACs, coffee machines and printers. This data is further used for querying the system and issue a ticket.

The IBM Watson Conversation API is used to build, test and deploy chatbot. It provides a platform to implement chatbot in following stages:

1. After creating a conversation service, a workspace is created for a chatbot application.
2. The intents are created. The intents specify the purpose of a specific idea. For each intent examples are added to train the Conversation for intent recognition.
3. Entities that specify context for the intents are added as well as synonyms for the specified entities are added.
4. Dialog flow is constructed which contains nodes defining steps in the conversation. The nodes with responses are created for each intent and entity based on the conditions they denote. When Watson natural language classifier recognises an intent or an entity, the response of that node is given as output of the chatbot.

The input is bound in JSON object and through REST API call sent to the chatbot plugin. Chatbot plugin sends the input to the IBM Watson Conversation API. After processing the input and generating response, it is captured in chatbot plugin. The response is checked in plugin and modified if necessary. Further, the chat application displays the final response.

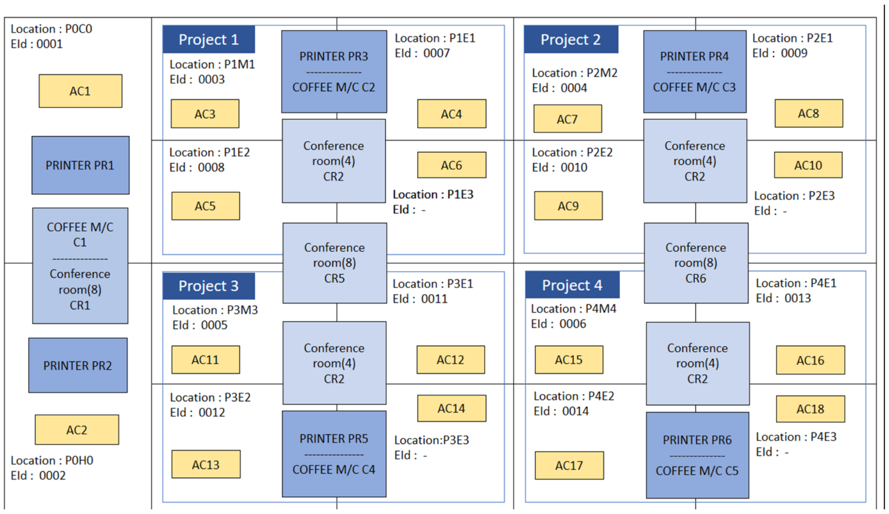
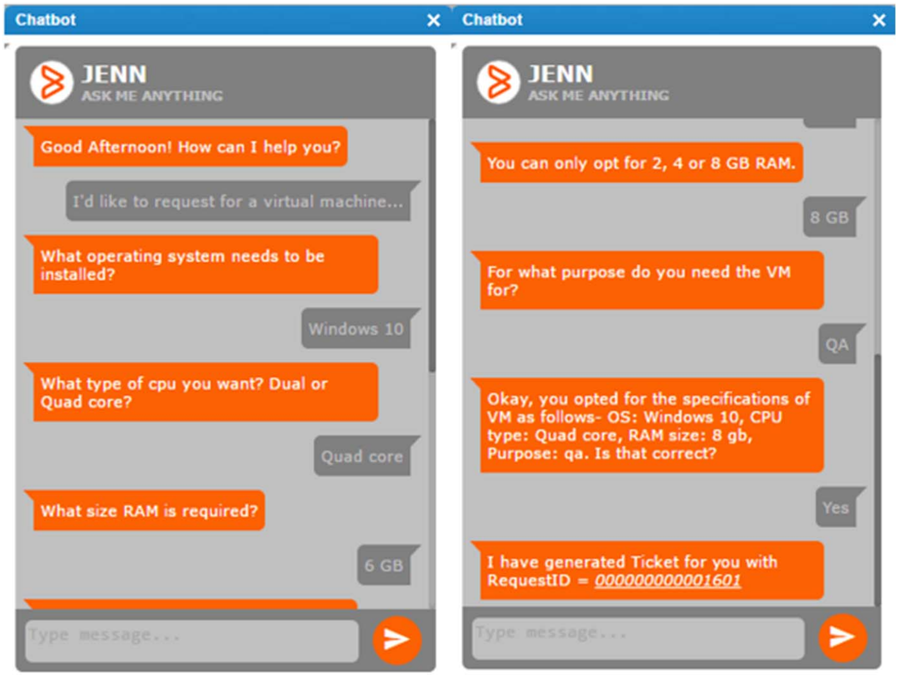


Fig. 3. Virtual Office Layout

The sample use cases implemented are as follows:

1. Installation query(printer/tomcat):
   - Suggest some solutions and generate ticket if query is not solved.
2. Location Change
   - User enters the new location to which he would like to move to.
   - The availability and the post feasibility of the location entered is checked.
   - Finally, the option for auto-allocate or ticket generation is given.
3. Coffee machine not working
   - Coffee machine near the user is identified. If not the same machine, the coffee machine ID is taken from the user.
   - Ticket is generated accordingly.
4. AC: Turn ON/OFF
   - AC near the user is identified.
   - If not the same AC, the other AC ID is taken from the user.
   - AC is turned ON/OFF accordingly.
5. Request Virtual Machine
   - Specifications for the virtual machine are taken from the user and accordingly ticket is generated.
6. Handle irrelevant requests.
   - Ticket is generated if irrelevant request is encountered more than once.
7. VPN not working
   - Suggest some solutions and generate ticket if query is not solved. Refer Fig. 4 for implementation of sample use case.



Good Afternoon! How can I help you?

I'd like to request for a virtual machine...

What operating system needs to be installed?

Windows 10

What type of cpu you want? Dual or Quad core?

Quad core

What size RAM is required?

6 GB

You can only opt for 2, 4 or 8 GB RAM.

8 GB

For what purpose do you need the VM for?

QA

Okay, you opted for the specifications of VM as follows- OS: Windows 10, CPU type: Quad core, RAM size: 8 gb, Purpose: qa. Is that correct?

Yes

I have generated Ticket for you with RequestID = 000000000001601

Fig. 4. Implementation of sample use case

V. Conclusion

In this model, we proposed a chatbot system using IBM Watson Conversation API and improve the experience of software firm's employees. The system would be capable of analysing natural language input given by the end user and respond back with the solutions in the dialogs. If the user is unsatisfied with the solutions provided by the chatbot, the ticket can be generated. The chatbot also stores the conversations related to each different query so that it can be attached as a chat history in case of ticket generation. Further, we could embed the chatbot in ITSM of various software companies.

Future work will regard the development of chatbot to handle attachments directly through the chatbot application and generation of answers implicitly.

References

[1] Vincenzo Di Lecce, Marco Calabrese Domenico Soldo, Alessandro Quarto, "Dialogue-Oriented Interface for Linguistic Human-Computer Interaction: a Chat-based Application", Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), 2010 IEEE International Conference.

[2] Niranjan.M, Saipreethy.M.S., Gireesh Kumar.T., "An Intelligent Question Answering Conversational Agent using Naïve Bayesian Classifier", 2012 IEEE International Conference on Technology Enhanced Education (ICTEE).

[3] SupratipGhose, JagatJoytiBarua "Toward the implementation of a Topic specific Dialogue based Natural Language Chatbot as an Undergraduate Advisor", Informatics, Electronics & Vision (ICIEV), 2013 International Conference.

[4] N T Thomas, "An E-business Chatbot using AIML and LSA", 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI).

[5] DivyaMadhu, Neeraj Jain C. J, ElmySebastain, ShinoyShaji, AnandhuAjayakumar, "A Novel Approach for Medical Assistance Using Trained Chatbot", International Conference on Inventive Communication and Computational Technologies (ICICCT 2017).

[6] BayuSetiaji, Ferry WahyuWibowo, "Chatbot Using A Knowledge in Database", 7th International Conference on Intelligent Systems, Modelling and Simulation, 2016.

[7] Vinyals, O., and Le, Q. V. 2015, "A neural conversational model", CoRR abs/1506.05869.

[8] [8] Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, Wei-Ying Ma, "Topic Aware Neural Response Generation", CoRR abs/1606.08340.

[9] Iulian V. Serban, Alessandro Sordoni, YoshuaBengio, Aaron Courville, Joelle Pineau, "Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models", CoRR abs/1507.04808.

[10] Kyunghyun Cho, Bart van Merrienboer, CaglarGulcehre, DzmitryBahdanau, FethiBougares, Holger Schwenk, YoshuaBengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation", CoRR abs/1406.1078.

[11] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, "Efficient Estimation of Word Representations in Vector Space", CoRR abs/1301.3781.

[12] P. Dhoolia, P. Chugh, P. Costa, A cognitive system for business and technical support: A case study, IBM Journal of Research and Development(Volume: 61, Issue: 1, Jan.-Feb. 1 2017).