International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

Customer Support Chatbot Leveraging Machine Learning

Hiral Paghadal[1], Anezka Virani[2], Apratim Shukla[3], Dr. G T Thampi[4]

[1,2,3]U.G. Students, Department of Information Technology, Thadomal Shahani Engineering College, Mumbai, Maharashtra, India
[4]Principal, Thadomal Shahani Engineering College, Mumbai, Maharashtra, India

---

Abstract – *Customer support plays an important role in an organization's ability to generate revenue and income. Support staff spend a lot of time answering questions via telephone to make sure that the customers are satisfied with their business. Customer support through telephone is time consuming, exasperating and possibly leaves the customer with unresolved issues. In this paper we introduce more efficient way to resolve customer queries. Today's customers have high expectations and want convenience, quick and accurate responses, complete and robust resolution, service that is available anywhere and anytime. All of these can be addressed with well-designed chatbots. The entire experience is conversational and chat is the most appropriate medium as it is quick and accurate. The aim is to implement a chatbot which can resolve customer queries, search the knowledgebase for resolution and give the solution. The chatbot will handle the queries; ultimately reducing the human effort.*

Key Words: **Rasa framework, Natural Language Processing, Rasa Natural language Understanding (NLU), Rasa Core, Artificial Intelligence, Machine Learning, Flask, Webhooks**

1. INTRODUCTION

Customer support and service is difficult to achieve. Customers buy products online, make payments, has queries related to products as a result they want good customer service for solving their queries. Traditionally, people use telephones to contact to the customer executive. This process is very time consuming as the customers need to wait on the line for a lot of time before their request is processed. The customers get frustrated when they ask the same question again and again, lodge complaints and they don't receive a response for days. Also, the cost of phone interaction between the customer as well as executive is also more. So, to solve this issue we introduce chatbots which is a computer program that we can talk to via text, chat or voice. Using Artificial Intelligence (AI) Powered chatbots, enterprises can be closer to achieving efficient and automated customer service which can lead to better engagement and understanding [1].

A chatbot is a computer program through which you can talk to, through messaging applications. The chatbot replies through the same messaging application, creating a back and forth conversation between the customer and the bot. The chatbot has the ability to respond immediately as they serve as round the clock agent which is available 24/7, 365 days. Chatbots reduces human error as well personalizes the customer service. Chatbots, are a major innovation in the field of AI.

Chatbots are highly responsive, interactive which resembles human conversations using AI tools and techniques and resolves customer queries or needs anytime with the ease of chat. A customer can put a question or query and the chatbot replies with the right response. Based on the situation, the chatbots can learn from the utterances in the conversation and further personalize the responses and learn from the past connections [1]. Chatbots have a lots benefits including a 24/7 customer service, timely responses and effective inquiry handling, reduced cost of customer service and best customer satisfaction. They outperform humans in terms of speed and accuracy.

2. LITERATURE SURVEY

Conversional assistants are becoming integral part of daily life. Rasa Core and Rasa Natural Language understanding (NLU) are easy to use tools for building conversational systems [2]. Rasa is an essential set of tools for building more advanced and efficient AI assistants/chatbots. The benefit of rasa is the infrastructure and tools which provides the user with high performance, resilient and proprietary intelligent chatbots that work. Rasa helps all developers create better text and voice-based chatbots. Rasa's NLU helps the developers with the technology and the tools necessary for capturing and understanding user input, determining the intent and entities. Rasa supports multiple languages, single and multiple intent, and both pre-trained and custom entities [3].

Rasa is an opensource framework for building AI bots. Rasa open source framework consists of two components: - Rasa NLU and Rasa core. Rasa recommends using both Rasa NLU and core, but they can be used independently of each other. Rasa core is the component which handles the dialogue engine for the framework and helps in creating more complex chatbots with customization. Rasa provides an opportunity for interactive learning. Chatbots can be enhanced because of the flexibility options provided by the Rasa framework. The chatbot can be easily deployed, integrated and connected to websites and applications [3].

Rasa being an open source framework it is very convenient and easy to customize. Most of the chatbot framework available are totally cloud based and provides software as a service. Business, enterprises and clients do not wish to share their data on cloud or any third-party service. Rasa fits the best when you don't want to send your data to external device. We choose rasa as our framework because it is not cloud based and can be easily customized. Rasa allows the user to build, host and deploy Rasa internally in our server or environment. Deploying the Rasa on our own server can help to secure the data. Rasa provides better control and flexibility in deploying the chatbot. It is free and open source which makes a go to choice for building chatbots. [4].

3. PROPOSED MODEL

In this paper we present a chatbot which is a banking Chatbot called Jarvis which resolves all the bank related queries. The chatbot's model can be divided into three sections – Backend, ML model and Frontend. The main functionality of the chatbot is carried out by Rasa Framework.

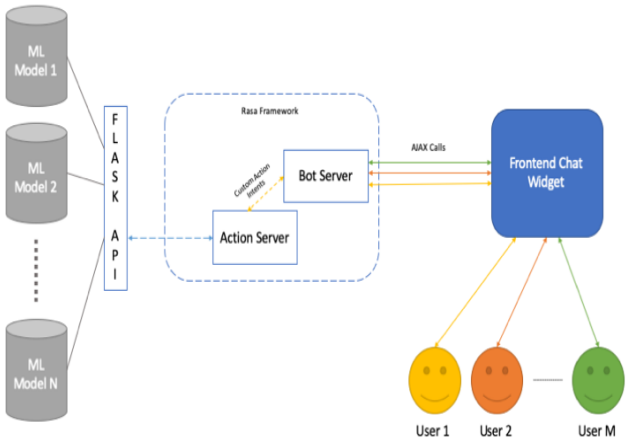

Fig -1: Proposed Model

3.1 Backend

The Backend is built on Rasa Framework. Rasa framework is based on Python. Rasa is responsible for handling the user input, identifying the intents and entities and creating the responses. We have used both Rasa NLU and Rasa core. Rasa NLU provides the capability for classification of intent and extraction of entity from the user input and helps in understanding what the user is saying [5]. Rasa NLU handles all NLP stuffs. Rasa NLU deals with teaching a chatbot on how to understand user inputs. The concept of intents is used by Rasa to describe how user messages should be categorized. Rasa NLU classifies the user inputs into one ore multiple intents. As soon as the user enters the query or question, Rasa receives the message from the end user, it extracts the "intent" and "entities" present in the message. Intent is what the user aims to say or what the user wants [6]. Suppose the user types "hey", "hello", "hi" the intent here is greet. Suppose if the user types "I want to block my debit card" or "I lost my debit card" the intent here is block_card.

Entity is extracting pieces of information from the user input, which helps the chatbot. It specifically helps to understand what a user is asking about by recognizing the structured data in the sentence. Suppose the user wants a loan. Then loan eligibility is the intent and the entities are gender, income, self-employed etc. are the entities. The intents and entities serve as training data for Rasa NLU. The training data contains multiple intents and entities for the chatbot to understand the text. Training data is written to nlu.md file. Each intent is followed by multiple examples of how user might express the intent. Multiple examples of customer queries are provided below each intent because people often make a spelling mistake or the statement is not grammatically correct. Mentioning many examples, even spelling mistakes can improve bot's capability to resolve the query. We need the above training data to train the NLU model. NLU model is used to extract the useful intent and entities from the text input. By training the NLU model on training data helps to identify the intent and entity.

A training pipeline is used to create an NLU model. A training pipeline is a sequence of processing steps which allows the model to learn the training data's underlying pattern. Once training data is ready, we need to feed it to the NLU model pipeline. Rasa has a number of different components which together makes a pipeline. All the components listed in the pipeline will get trained. The input and query data flows through the pipeline for intent classification and entity extraction. The supervised_embedding pipeline trains the model from the starch using the training data. It has many advantages like adapting to domain specific words since the model is trained on out our data. It allows to build assistants in any language and also supports multiple intents. The processing pipeline is defined in the config.yml file [7].

Rasa has excellent command line interface. We now run the command "rasa train nlu". This command trains the model and saves it in the model's directory. We can now test the model using "rasa shell nlu" command. The bot can now understand what user says. The bot now needs to give a response. This is where dialogue management comes into picture. To accomplish this end to end conversation we use rasa core the second component of rasa. Rasa core teaches the bot to respond to the query. Dialogue management controls the next action the bot takes during the conversation. Based on the intents, entities and conversation history, Rasa core decides which text response should be sent to the user. Rasa core uses Machine earning to pick up conversation patterns from conversation examples. Based on these patterns, the bot

can generalize which allows the model to predict the next best action. We need to provide training examples containing a few conversations, it is not necessary to provide every possible conversation since it gathers new conversational data directly through real user interactions.

Stories is the basic unit in rasa core for dialogue management and training. Stories are sample conversations between user and bot. It is stored in stories.md file. Stories have a specific format to represent the input as intent, entities and responses are the action names [8]. Actions is operations rasa runs for replying to the user query. Actions also contains custom code which retrieves a response by making an API call [9]. There are two types of actions utterances and custom actions. Utterances are replies intended towards a specific intent. Custom action executes a custom code like fetching data from API. Defining a domain file is important for dialogue management. The domain file includes what the user intents, entities, Reponses the model provides such as utterances and custom actions and slots. Slots are important element of domain file. Slots functions as the bot's memory. It is used by the bot to remember important details. Slots act as key value pair. There many types of slot. Some of them are text, categorial, float and list. In our chatbot Jarvis the bot predicts the loan legibility. For that the bot queries the customer details. We have categorical slots for intents like gender(male, female), selfemployed (yes, no), education (yes, no).

Custom action can check if the user is eligible for loan or not. Custom action runs on a separate server than the server that the model runs on. When a custom action is predicted, Rasa calls an endpoint. This endpoint should be a webserver that reacts this call, runs the code and returns the information to modify the dialogue state. The action server is deployed on localhost:5055 using webhooks. We now need to retrain the model using the command "rasa train". The chatbot can be accessed using a POST API call with user id and user message in the body in the json format and returns the user id and the bot message in json format. Backend is such that any ML model can be deployed on the server and can be used by making an API call. Rasa also provides interactive learning, in which you can provide feedback to the bot while talking to it. This is a powerful way to explore what the bot can do, and the easiest way to fix any mistakes it makes.

3.2 ML Model Integrations

The loan is one of the most important criteria to any customer. Banks need effective business strategies to influence more customers to apply their loans. However, there are some customers who are not able to pay off the loan after their application are approved. Therefore, many Financial institutions take several variables into account while approving a loan. **Our machine learning model predicts whether you are eligible for a loan or not** based on whether a given borrower will be able to fully pay off the loan.

ML model can be integrated with the chatbot and deployed on server. We can integrate any number of ML model depending on the need of the chatbot. ML model here is used for loan prediction, which determines whether you are eligible for a loan or not based on entities like gender(Male or Female), Married (Yes or No), Self-Employed(Yes or no), Number of dependents(0/1/2/3+), Credit History(0-bad / 1-good), Property Area(Urban/Semi-Urban/Rural), Loan Amount(Required), Loan Tenure in years(Preferred), Applicant income, Co-applicant Income, EMI (calculated), Total Income. To remove the skewness in the graph of total income we have used a log function to smoothen the graph. A csv file of records is taken, the data is cleaned and the missing values are dealt. The categorical values are replaced by mode, numerical values are replaced by median and the categorical values are mapped to 0/1.

We have used logistic regression for Loan predictability. Logistic Regression is a binary Predictive model. The purpose of this algorithm is to find the plane that separate two types. We used model selection to choose between Random Forest, Logistic Regression, Naïve Bayes. Logistic had the Highest accuracy of 80%. Based on the above entities the model determines if a particular user is eligible for loan or not and responds with a yes or no. The model is deployed on localhost:5000 using Flask API. Flask API accesses models to make the prediction and sends the prediction to the chatbot in JSON format.

3.3 Frontend

The frontend for our implementation is a chat widget. The widget can be easily added to the HTML code for any website. It has been made using HTML, CSS and JavaScript. The widget is responsive and resizes depending on the viewport size of the screen that it is being displayed on. It makes use of images to represent the user and the bot. The color and images of the widget are easily customizable. Rasa allows the use of UI components like buttons and images to be used as response by the bot. These can be defined in the domain.yml file under 'utterances'. As shown in Fig-2 the JavaScript code identifies these UI components used in the bot's response and renders them correctly.

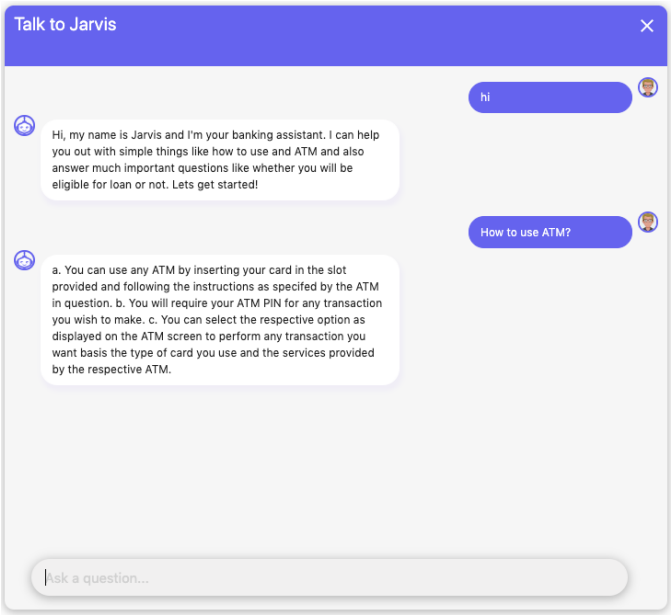

Fig -2: A text-only bot response

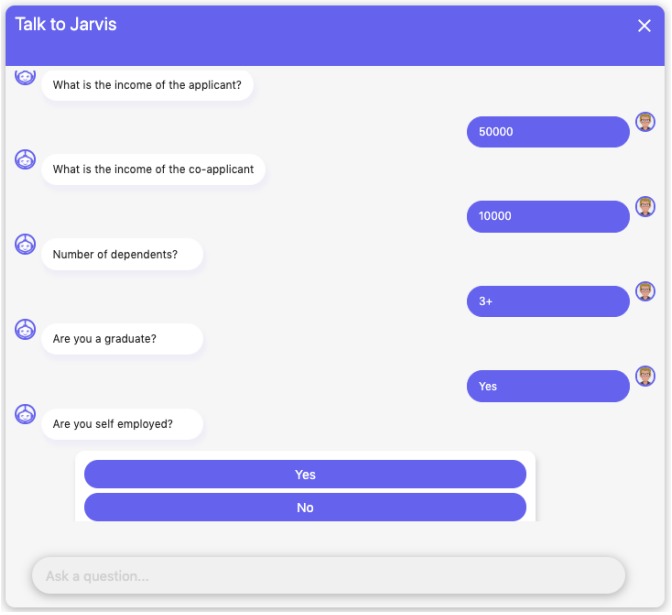

Fig -3: Buttons for response

4. RESULT AND ANALYSIS

The proposed model is scalable and secure. Since the chatbot can be deployed on internal servers, it eliminates the scare of data loss or interception of data. Rasa makes use of natural language processing libraries like spaCy. These libraries can detect and process spelling errors, abbreviations, etc. This makes the chatbot tolerant to users' mistakes and handles common language conventions seamlessly. The ML models are deployed such that they are accessible via a Flask API. Multiple models can be integrated to the chatbot. Each model can be defined as a separate intent(s) in the chatbots domain. These models will be accessed through custom actions defined in the chatbots domain. Thus, scalability can be achieved in terms of the number of models that can be integrated with the chatbot.

5. CONCLUSION

Chatbots are becoming an integral part of the digital world. It is necessary that the customer needs are addressed as well as customers are satisfied through the business. Customer expectations are growing with increasing technological development. Customers satisfaction is very important to businesses and enterprises because if the customers are not satisfied with the service customers never return. Chatbot has the ability to solve most of the service-related business problems. Customer service has always been critical for the determination of the success for any business organization. Chatbots enable a good customer service that never sleeps. Chatbots are powerful and dynamic providing a wide range of services. Chatbot development is a very important topic in AI industry and matter of research today. Chatbots are still considered as emerging technology, but they are quickly maturing and becoming an important factor for business organizations. In a nutshell chatbots are an innovative way for humans to interact with the software. Chatbots may revolutionize the way the customer service works.

REFERENCES

[1] White paper by Infosys on Power through AI and Automation through Chatbots
https://www.infosys.com/services/microsoft-dynamics/Documents/AI-Automation-Chatbots-Web.pdf

[2] Rasa: Open Source Language Understanding and Dialogue Management by Tom Bocklisch, Joey Faulkner, Nick Pawlowski, Alan Nichol
https://arxiv.org/pdf/1712.05181.pdf

[3] Website of Rasa
https://rasa.com/product/why-rasa/

[4] How to build a chatbot with Rasa: Complete Guide
https://www.datasciencelearner.com/how-to-build-a-chatbot-rasa-complete-guide/

[5] https://blog.rasa.com/the-rasa-masterclass-handbook-episode-1/

[6] https://blog.rasa.com/the-rasa-masterclass-handbook-episode-2/

[7] https://blog.rasa.com/the-rasa-masterclass-handbook-episode-3/

[8] https://blog.rasa.com/the-rasa-masterclass-handbook-episode-5/

[9] Build a Conversational Chatbot with Rasa Stack and Python – Rasa NLU by Romil Jain
https://medium.com/@itsromiljain/build-a-conversational-chatbot-with-rasa-stack-and-python-rasa-nlu-b79dfbe59491