CHATBOT WITH WATSON

Abstract :

Chatbots are the most important but still evolving feature for a website. This technology is also getting integrated with new generation hardware devices to improve autonomous tasks and reduce human effort. The integration of a chatbot with current other applications is possible because of recent research on ”human parity level speech detection” and smarter sentiment analysis. Traditionally, actual humans were operating chatbots after sending the first default message to the end-user. Nevertheless, now, many approaches can make these chatbot implementations easy for developers. There are two types of chatbots, the first type is domain-specific, which is also called machinedriven dialog systems. In this type of system, the end-user needs to follow the instructions given by machine, and this type of system can handle only some domain-specific scenarios, and the second type of chatbot is a general-purpose chatbot that can handle various domains at the same time. In the current era, there are various approaches available for developers to implement a chatbot, for example, machine learning approaches that include semantic parsing, tone analyzer, etc. or Rulebased approaches that we will discuss in detail in next sections. To create a chatbot, the developer needs various tools to handle the user’s input and process it into the required output. There are many open-source and paid systems available in the market, some of the open-source systems are Microsoft Bot Framework, Rasa, Botpress, ANA Chat, and some of the paid systems are IBM Watson, Amazon Lex and many more. This paper explains the methods to create a basic chatbot using one of the various approaches to creating dialog systems. It includes the explanation of various architectural tools required for creating a chatbot system.

Various approaches to create dialog systems :

Rule-based approach Rule-based systems use rules for knowledge representation.

The basic idea of Rule-based algorithms is the hierarchy of rules that govern how to transform user input into output dialogue or actions. Rule-based system is the easiest method to create a chatbot. Rules can be simple or complex, depending on the requirement of the chatbot system. However, these chatbots lack functionality in case rules are not able to map themselves with user input. The best example of this type of system is Eliza, a chatbot created at MIT in the 1960s. This system was able to assign a unique id to each keyword and then reassembled the input sentence based on the highest-ranking keyword. In Figure 1, the Rule-based system tries to parse the user input to get the keywords required for the next action.

Retrieval-based approach

Retrieval-based is the most common dialog system that is being used in almost the majority of chatbots in the market today. After getting user input, this system tries to find the adequate response of Figure 1: Example of ”Rule based approach” Figure 2: Architecture of ”Retrieval based approach” that input from the database that included a predefined set of responses. E.g., in Figure 2, the question is analyzed and processed, then the answer is generated from the answer database. The next step is to score the answers based on evidence sources. The final scoring of answers is the last step in the example shown. This system uses a heuristic approach to retrieve data from the database. The retrieval-based approach is the same as the prediction problem in which a predefined template is used to determine the response by using keyword matching. Retrieval-based systems also can include some

complex algorithms like deep learning or machine learning algorithms to retrieve data. Original IBM Watson was built on this approach. However, there are many disadvantages to these kinds of systems as it is complicated to scale dialog.

Figure 1: Example of ”Rule based approach”

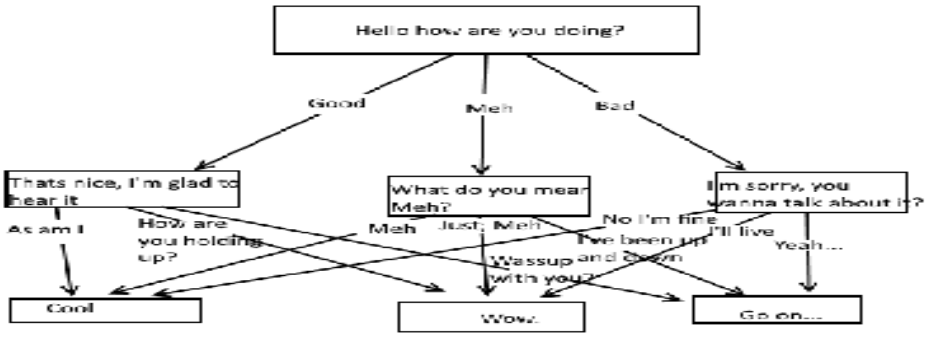


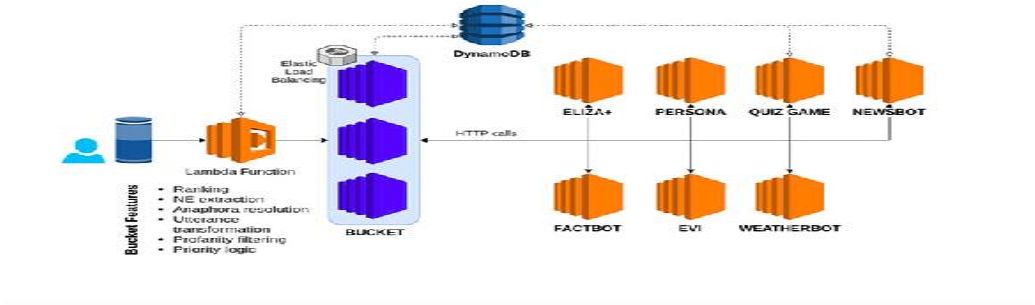
Figure 2: Architecture of ”Retrieval based ap- proach”

Generative methods :

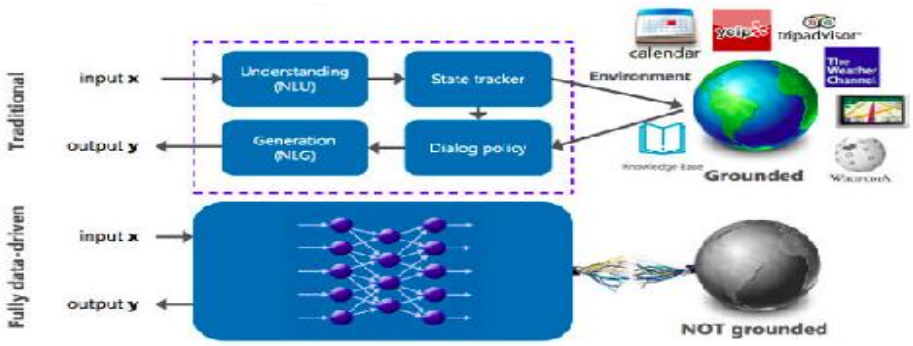
There is a problem with previously mentioned methods as these

models do not generate new content by themselves. So this method overcomes this issue by using Artificial Intelligence methods to generate new content instead of using pre-defined responses. Model training requires a large amount of data for training. Supervised learning, reinforcement learning, and adversarial learning are the techniques developers can use in dialog systems. Figure 3, shows the architecture of Supervised learning with Reinforcement learning and is defined by the 5-tuple (S, A, P, R, T); a set of states S and a set of actions A, P is the state transition probability, reward function R: S ? R and T is dis- counted rewards. Figure 4, the architecture of ad- versarial learning, shows that there is a real-world conversation database in the server, and when the user inputs the query, that query gets searched in the real world conversation, and then discriminator filter out the possible solutions for that query. Se- quence to sequence modeling problem is the prob- lem in which input from the user has the same se- quence of output, for example, machine transla- tion, in which the system tries to translate one lan- guage to another, but input sequence length can- not always be the same as output sequence length. If we use Reinforcement learning with Supervised learning, then we can remove the sequence to se- quence problem as this prioritizes high-priority, high-probability response content. In that type of system, it is hard for noun classes to be considered in output as proper nouns are less frequent. Now, if we divide the training into two phases first, one is the observational phase in which we use Super- vised learning on existing dialogues to imitate hu- man behavior. The second phase is a trial and error phase in which we use Reinforcement learning to add new situations and dialog inputs that were not present in training data. Adversarial learning can also improve neural dialog output. In this method, the agent learns by using a small Turing test on one side to create human-like responses and discrimi- nator network judges to check whether this input is from actual human or the computer.

Interactive learning



Language is highly interactive. Interactive learn- ing is only a developing research area. In In- teractive learning method, human knows the tar- get but do not know how to reach that target or do not know the methods to achieve that as well. The computer system has control over the methods only, but the computer does not understand the hu- man language. This method is based on the map- ping of human language with tasks. To reach the goal, the human is trying iterative steps and map- ping the human language with actions performed by a machine



Microsoft Bot Framework:

Microsoft Bot Framework is an open-source product of Microsoft to create chatbots. It contains three services; Bot Builder SDK, Bot Framework Figure 8: Components of a conversational AI experience portal, and channels. Bot Builder SDK is the essential toolkit that is used to create dialog systems for the chatbot. However, the main drawback of this system is that this SDK comes only in javascript and dot-net. Microsoft Bot Framework provides multiple services to create a chatbot like a language that plays the part of NLU, and another new service is QnA Maker. This service is much similar to the Discovery feature of IBM Watson that we will discuss in further parts. A simple question and answer chatbot can be created without using many complex algorithms or just by using this feature. In this feature, the user needs to upload a structured document containing FAQs. The decision is the feature used to determine the next step in architecture. It helps in making an efficient decision for the generation of output. Figure 8 shows the architecture of the Microsoft Bot Framework. It includes cognitive services. Azure Bot Service includes hardware device output, and tones, speech, type words acts as an input channel where dispatch acts as output channel after input processing. The output can be used as third party input or to start the new action for further processing. 4 IBM Watson Watson was named after the first CEO of IBM, Thomas J. Watson. Initially, this system was generated to answer questions to quiz show Jeopardy in 2011. In 2013 IBM Watson was available to the public for commercial use. IBM Watson was created as a question answering computing system to apply machine learning, information retrieval, automated reasoning, knowledge representation, and natural language processing in the field of open domain question answering. Back then, it was just a question-answering machine, but now afterrecent developments in IBM Watson the system can perform various actions in ’see,’ ’hear,’ ’read,’ ’talk,’ ’taste,’ ’interpret,’ ’learn’ and ’recommend.’ The combination of Java, C++, and Prolog was used to create a commercial product of IBM Watson in the beginning. It runs

on SUSE Linux Enterprise Server 11 and uses Apache Hadoop to provide distributed computing. It contains a big database of information; it includes data from encyclopedias, dictionaries, thesauri, newswire articles, and literary works. IBM Watson also provides millions of documents to create a knowledge base for application use.

IBM Watson Developer Cloud

IBM Watson Developer Cloud comes in three models. 4.1.1 Shared Shared is a cloud deployment of WDC deployed on Bluemix Shared. It offers robust data security in a multi-tenant environment and encryption of data in transit and at rest. 4.1.2 Premium Premium is a single-tenant virtual environment deployment of WDC deployed on Bluemix Shared. Premium adds isolated compute, which is also the security features of Shared. 4.1.3 Dedicated Dedicated is a private cloud deployment of WDC built on top of Bluemix Dedicated. The Dedicated deployment addresses data security and regulatory compliance requirements by offering hardware isolation and data encryption, hosted in a SoftLayer data center. 4.2 Tools IBM Watson has multiple tools to make the chatbot system works. One of the modules in Watson is Understand, which helps in understanding imaginary or unstructured data like humans understand. The reason is other tools that help in making form hypotheses and grasp underlying concepts. Learn is another module which helps in interactive learning so that new outcomes can be improved. Interact is the module helps in making chatbot interact like actual humans with abilities to talk, see, and hear. 4.2.1 Watson Assistant According to IBM Watson, ”Watson Assistant” is more than a chatbot. Its essential capability is that it knows how to unify various channels to perform various actions for chatbot users. It knows when to search output from a knowledge base or when to ask for more detail from the user or when to represent something to human users. Watson’s assistant can run on any cloud or server, allowing developers to create a basic structure of

an AI-based chatbot system. Assistant store data with sessions so that the system can have user interaction saved for future sessions. It does not matter if the request is complex or straightforward; the assistant will break the request and map the various actions for subatomic components of request. It gets better with time as this service is based on the interactive learning mechanism of dialog systems. The best advantage of this service is that it is effortless to deploy.

System Created :

Architecture of System I created a system to analyze the output of current IBM tools to perform some improvements in the tools. I used above mentioned four tools for the system. I got the necessary structure to create chatbot from IBM Watson demo application. I altered the database and some files which were used to train the discovery model for the chatbot. The minimum score required to become output from the question-answer file was configured to updated and edited the intents and dialog flow in the UI portal of IBM Watson. I created a basic banking advisor chatbot system. Node.js was used as IBM Watson SDK, and React.js was used as the chatbot user interface. The communication between these channels was implemented using the REST framework. Every module from IBM Watson was plug and play. Developers can also add another tool to enhance the functionality of the chatbot system. Figure 9 explains the architecture of the system I created. It includes vital tools necessary for creating a chatbot. It also describes the file input and output parameter for the discovery module.

Future Implementation :

For the tone analyzer, we can add more types of emotions like abusive text or relations between subject and object. Slang is essential to make the system work. However, the different areas can have different slang words, so other improvements can be, adding these features using cloud computing and analyzing the position of the user and using an appropriate distributed network model to handle requests instead of one central database. There is also a lot more space to improve the discovery feature of IBM Watson. Improving result relevancy is a never-ending process. An improvement over connections in organizations, concepts, relationships, locations can be made.