Creating domain specific chatbot using IBM Watson

Himanshu Bansal
University of Tuebingen

Creating domain specific chatbot using IBM Watson

Himanshu Bansal
himanshu.bansal@student.uni-tuebingen.de

Abstract

The main aim of this paper is to show how to use IBM Watson tools to create a domain-specific chatbot and figure out the importance and effectiveness of some tools by IBM Watson to evaluate their accuracy. This paper also tries to explain the advantages and disadvantages of various tools also showing some methods that can be used to improve the chatbot system or particular modules of the chatbot.

1 Introduction

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 machine-driven 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 Rule-based 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.

2 Various approaches to create dialog systems

2.1 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.

2.2 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

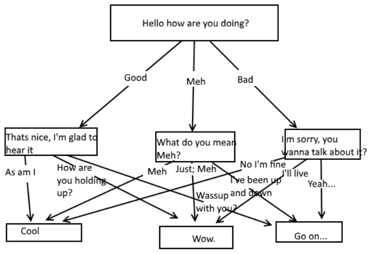


Figure 1: Example of ”Rule based approach”

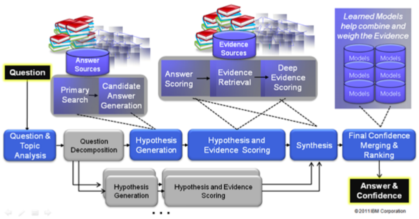


Figure 2: Architecture of ”Retrieval based approach”

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.

2.3 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 discounted rewards. Figure 4, the architecture of adversarial 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. Sequence to sequence modeling problem is the problem in which input from the user has the same sequence of output, for example, machine translation, in which the system tries to translate one language to another, but input sequence length cannot always be the same as output sequence length. If we use Reinforcement learning with Supervised learning, then we can remove the sequence to sequence 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 Supervised learning on existing dialogues to imitate human 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 discriminator network judges to check whether this input is from actual human or the computer.

2.4 Ensemble methods

If we want to make a dialog system that can talk about anything that means the system that is not topic or domain-specific, then this type of model comes handy. This model is a combination of generative method, rule-based, and retrieval-based. As we can see in Figure 5, Lambda functions support the communication between buckets that contains data with various chatbot assistants using an HTTP call.

2.5 Grounded learning

Chatbots do not have knowledge as humans have. To be a personal conversational partner, the chat-

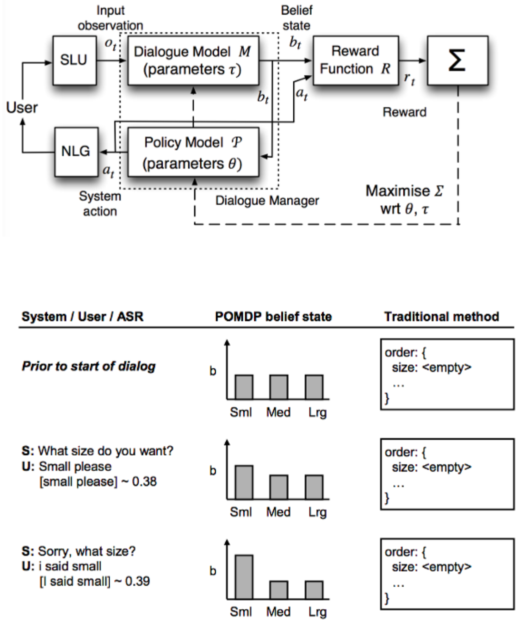


Figure 3: Architecture of "Supervised learning with Reinforcement learning"

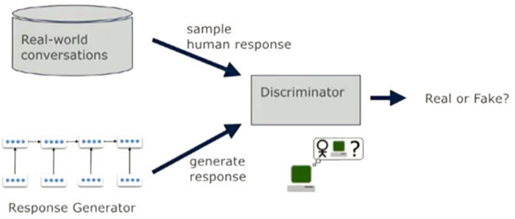


Figure 4: Architecture of "Adversarial learning"

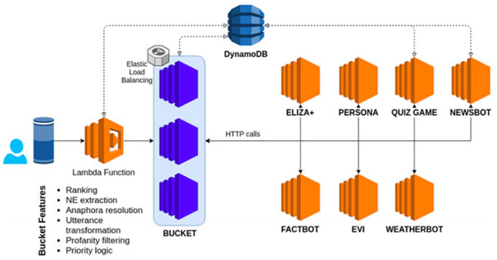


Figure 5: Architecture of "Ensemble methods"

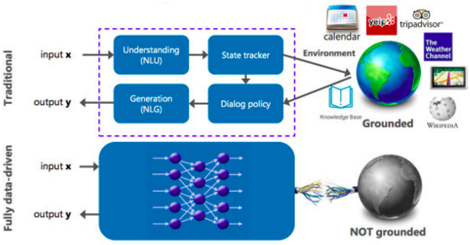
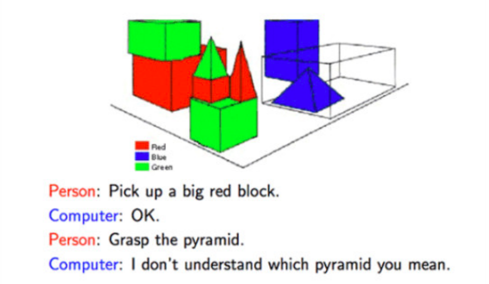


Figure 6: Architecture of "Grounded learning"

bot needs to understand the human and also need to possess knowledge about human behavior. This kind of knowledge cannot be put to the dialog system by training or any machine learning algorithms. Other approaches lack the functionality where it comes to logical thinking. Grounded learning is a new area of research. In this type of method, an assignment is divided into atomic subtasks, and then the neural network tries to figure out the solution of each task but also maintaining the interdependency or relation of subtasks. Human conversations also depend on the context of the dialog. So to create a fully-functional method, the neural network in grounded learning needs to access end to end context. Figure 6 explains two types of learning; Traditional and Fully data-driven. Traditional systems input passes through various sub-modules of Natural Language Processing but also connected to the module, which contains data about the open world. It can be Wikipedia, calendars, GPS, and so forth. But contrary to this system. Entirely data-driven systems are based on the sequence to sequence learning, but sometimes predefined data is not enough to produce the required output.

2.6 Interactive learning

Language is highly interactive. Interactive learning is only a developing research area. In Interactive learning method, human knows the target 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 human language. This method is based on the mapping of human language with tasks. To reach the goal, the human is trying iterative steps and mapping the human language with actions performed by a machine. Figure 7 explains the interactive



Person: Pick up a big red block.
Computer: OK.
Person: Grasp the pyramid.
Computer: I don't understand which pyramid you mean.

Figure 7: Example of ”Interactive learning”

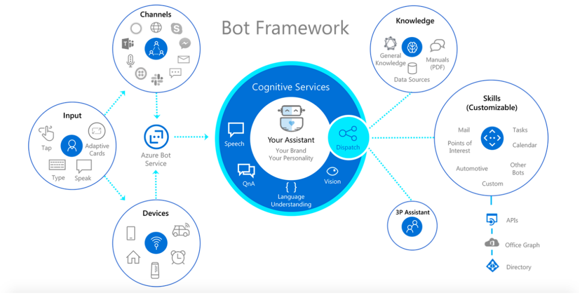


Figure 8: Components of a conversational AI experience

learning using an example, Person asks for the big red block; this information includes everything required to select one of an object out of the list. However, in the next user input, the user asks for the pyramid, but there are three pyramids in the list so, the computer asks for the next piece of information from the user to filter out the required information from available data.

3 Open Source Systems

3.1 Rasa

AI assistants and chatbots. It uses two main modules; Natural language understanding and Core. Natural language understanding is used to get useful information from user input, and Core is used to hold conversations and decide what the next step in the chat is. In this type of system, architecture is quite simple. The message then received and passed to an Interpreter, that extracts the original text and decide the intent or any intent that was found in the text. This part is handled by Natural language understanding. There is a module called tracker, which keeps track of the conversation. It handles the information about the new message or output generated by the NLU unit. After that, policy receives the current state of tracker; that's how it selects the next action to be performed. After this, the user sent a response. Rasa supports multiple languages. Developers can use any language to train a model for the chatbot. Rasa contains pre-trained word vectors, which helps the user to start the chatbot creation with less data input at the start.

3.2 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 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 after recent 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.

4.1 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.

4.2.2 NLU

NLU stands for Natural Language Understanding. It is getting used with other tools to make chatbot run with some analysis of semantic features like text input, including categories, concepts, emotion, entities, keywords, metadata, relations, semantic roles, and sentiment. It categorizes the content into a five-level hierarchy. It also supports various languages. It tries to create a relation between object and subject with analysis of the period. It performs actions after sentiment analysis of content. For, eg. "The Nobel Prize in Physics 1921 was awarded to Albert Einstein.". After using this NLU tool on this sentence, the output in terms of the relation is "awardedTo" relation between "Noble Prize in Physics" and "Albert Einstein" and "timeOf" relation between "1921" and "awarded.".

4.2.3 Discovery

It is a potent tool used to extract information from documents offline but very fast. For a developer analyzing the unstructured data are the most time consuming and challenging task. Massive amounts of diverse and dispersed unstructured content NLP APIs are arduous to integrate with dialog systems. Here this module comes handy; Discovery uses very sophisticated algorithms to make use of unstructured data for enrichment of required text. It supports multiple file types. It performs boolean, filter, or aggregation queries to

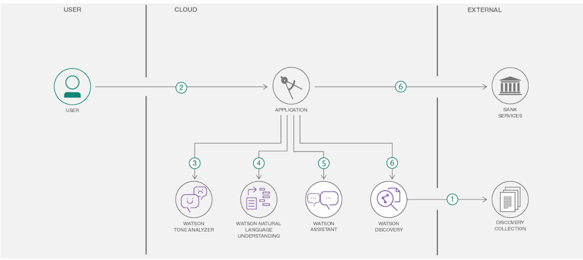


Figure 9: Architecture of System Created as Demo

discover matching patterns or answers. It includes advanced natural language processing steps to extract intents, entities, relations, or sentiments. It is a feature of cognitive search capability. It gives the solution to existing questions that need their solution from the document file. The data is in the form of Rest APIS.

4.2.4 Tone Analyser

The other most crucial task of the dialog system is to detect emotional tones in written text. This service is useful as it does the analyze at sentence or document level. Businesses can get a review of their products just by getting emotional tones from the text. It uses deep linguistic properties to check the tone of the message. The output of the request is JSON response that is created by a combination of 13 different tones grouped by an array with the score value given to each tone.

5 System Created

5.1 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 5. I 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.

6 Role of Cloud in modern development

Nowadays, cloud computing enables the new breed of applications. They are defining some of the benefits of cloud-like low latency, location-specific programs to get an appropriate audience, mobility, or active presence of real-time applications. ”Pay as you use,” is the most cost-effective way to create applications nowadays. If we integrate cloud computing with dialog systems, we can create a great working model of chatbot system for modern applications. Developers need not take care of the configuration of servers to make chatbot work. Some of the great examples of serverless computing nowadays are Amazon web services or Microsoft Azure. Writing a wrapper for IBM Watson tools and Rest API’s in lambda functions can save a significant amount of time and resource power. This makes it easy for an application to scale in case of increasing load.

7 Discussion

The tone analyzer module is created by using the supervised machine learning algorithm for semantic parsing. The manually annotated training set trained the model. The training data was taken from social media posts; thats the reason this tone analyzer service can recognize contextual emotions based on communication that occurs on social media. This service handles the negation pretty well. Another great benefit of using these tools is that these are easy to deploy independently. If a new developer wants to remove one feature and use another according to the requirement of business workflow, it is effortless to configure the system settings. The quality of models used in these tools is getting improved every year.

8 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.

References

[1] Hall, P.D. and Venigalla, V. and Janarthanam, S. *Hands-On Chatbots and Conversational UI Development*. Build chatbots and voice user interfaces with Chatfuel, Dialogflow, Microsoft Bot Framework, Twilio, and Alexa Skills Packt Publishing.

[2] Yan M, Castro P, Cheng P, Ishakian V. (2016) *Building a chatbot with serverless computing.*. InProceedings of the 1st International Workshop on Mashups of Things and APIs ACM.

[3] Rasa Support NLU *Building contextual chatbots and AI assistants* . Language Understanding for Chatbots and AI assistants https://rasa.com/docs/rasa/nlu/about/.

[4] Rasa Support Responses *Retraining the bot just to change the text copy can be suboptimal for some workflows. Thats why Core also allows you to outsource the response generation and separate it from the dialogue learning.* . https://rasa.com/docs/rasa/core/responses/.

[5] Rasa Support Knowledge Base Actions *Building contextual chatbots and AI assistants* . https://rasa.com/docs/rasa/core/knowledge-bases/.

[6] Rasa Support Dialogue Elements *Rasa NLU is an open-source natural language processing tool for intent classification and entity extraction in chatbots.*. https://rasa.com/docs/rasa/dialogue-elements/dialogue-elements/.

[7] Mariya Yao *6 Technical Approaches For Building Conversational AI*. https://www.topbots.com/building-conversational-ai/.

[8] Microsoft Azure Bot Service *Azure Bot Service provides an integrated environment that is purpose-built for bot development, enabling you to build, connect, test, deploy, and manage intelligent bots, all from one place.*. https://azure.microsoft.com/en-in/services/bot- service/.

[9] IBM Watson - Natural Language Understanding *Natural language processing for advanced text analysis.*. https://www.ibm.com/watson/services/natural-language-understanding/.

[10] IBM Watson - Watson Assistant *Watson Assistant is IBMs AI product that lets you build, train, and deploy conversational interactions into any application, device, or channel.*. https://www.ibm.com/cloud/watson-assistant/.

[11] IBM Watson - Watson Discovery *Watson Discovery is an award-winning AI search technology that eliminates data silos and retrieves information buried inside enterprise data.*. https://www.ibm.com/cloud/watson-discovery.

[12] IBM Watson - IBM Watson Machine Learning *IBM Watson Machine Learning helps data scientists and developers accelerate AI and machine learning deployment.* . https://www.ibm.com/cloud/machine-learning.