Smartbot: A New Way of Learning

A Project Work
submitted in partial fulfillment of the
requirements for the degree of

Bachelor of Technology
in
Computer Science and Engineering

Submitted By
Ayushman
(1513101134)

Under the supervision of
T. Edison for Dr. R. Viswanathan
Professor

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA – 201306
MAY 2019

DECLARATION

Project Title: **Smartbot: A New Way of Learning**

Degree for which the project work is submitted: **Bachelor of Technology in Computer Science and Engineering**

I declare that the presented project represents largely my own ideas and work in my own words. Where others ideas or words have been included, I have adequately cited and listed in the reference materials. The report has been prepared without resorting to plagiarism. I have adhered to all principles of academic honesty and integrity. No falsified or fabricated data have been presented in the report. I understand that any violation of the above will cause for disciplinary action by the Institute, including revoking the conferred degree, if conferred, and can also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been taken.

Ayushman
Enrollment No. 1513101134

Date: 30th April 2019

CERTIFICATE

It is certified that the work contained in this project entitled **"Smartbot: A New Way of Learning"** submitted by **Ayushman (Enrollment No. 1513101134)**, for the degree of Bachelor of Technology in Computer Science and Engineering is absolutely based on his/her own work carried out under my supervision and this project work has not been submitted elsewhere for any degree.

T. Edison
Assistant Professor
School of Computing Science and Engineering
Galgotias University
Greater Noida, UP, India

Date: 30th April 2019

Countersigned by

Prof. (Dr.) Sanjeev Kumar Pippal
Professor and Associate Dean
School of Computing Science and Engineering
Galgotias University
Greater Noida, UP, India

ABSTRACT

Artificial intelligence (AI) can move one step ahead in the education sector. The AI-enabled adaptive software can predict test performance, scores and provide solutions to long-term learning problems. Teachers can leverage e-learning portals to provide instructions, take tests and give feedback. Artificial intelligence can be a driving factor in this field. As we can see that 'Artificial intelligence' is an emerging and indeed a very powerful top-notch technology, so we should use this 'Silver Bullet' to solve the critical and sophisticated problems of our society, of our nation, and so on. The most desired field and the field that I think we all should focus on is Indian Education System. As we know that, Technology is changing at a very high pace but the irony is that our education system is not up to date with it at all. We are using the same methodologies and the same tactics and tools for a very long time. As it is said that '*Old ways will not open new doors*' and as the George Bernard Shaw said Progress is impossible without change and those who cannot change their minds can't change anything. As we can see that today's education system mainly focus on mugging-up dates and facts. It does not have to do anything with educating the mind or we can say creativity. It does not teach the students, how to face and overcome the challenges of real world and problems of their personal life. Our AI-bot will help the students in achieving "*Professional as well as Personal Success*". It will act as personal coach and mentor for students.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my guide, T. Edison his valuable guidance, consistent encouragement, personal caring, timely help and providing me with an excellent atmosphere for doing research. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to me for completing this project work.

Ayushman

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

I am trying to improve the common problems faced by students and other individuals who are lost in making some important life decisions. Our bot presents them several paths according to their interests and also is a very good conversationalist for some students who are in stress due to any reason.

Chat bots were one of the first types of automated programs to be called "bots" and became popular in the 1990s, with the rise of online chatrooms. These bots are scripts that look for certain text patterns submitted by chat room participants and respond with automated actions. For example, a chat bot might warn a user if a person is using inappropriate language. So, chatbot are playing a very significant role in today's world. Our chatbot model is different in many aspects as compared to the bots we generally see in our day to day life. This bot will be able to help students of a particular category which are having some particular questions in their mind. Questions can be of many domains such as personal, academics and psychological. This bot will also be able to help parents as well as teachers because there might be some cases in which a parent or a teacher might be in a difficult situation when it comes to handling a child. It will suggest best possible solution for a particular problem at a given time. For example: A student can ask several questions such as How to start career in machine learning? Our model provides a real-time question-answering, engaging and adaptive learning experience for students, teachers and parents.

1.2 Hardware Requirements

Our bot doesn't require any specific hardware requirements because it can easily run on any platform all it just needs is an internet connection.

- Smartphone (Android, iOS, Windows)
- Laptop or PC

1.3 Software Requirements

- Slack
- Browser

1.4 Language Support

My bot can be trained in many languages, I'm using English as my primary language in my bot. Natural Language Understanding supports a variety of languages depending on which features you analyze. Currently, English is the only language that is supported across all features. The rest of the languages have limited support. Some of the main languages that our bot can be trained on are as follows:

- Arabic
- Chinese (Simplified)
- Dutch
- French
- German
- Italian
- Japanese
- Korean
- Portuguese
- Russian
- Spanish
- Swedish

CHAPTER 2

IMPLEMENTATION

2.1 Implemented Model

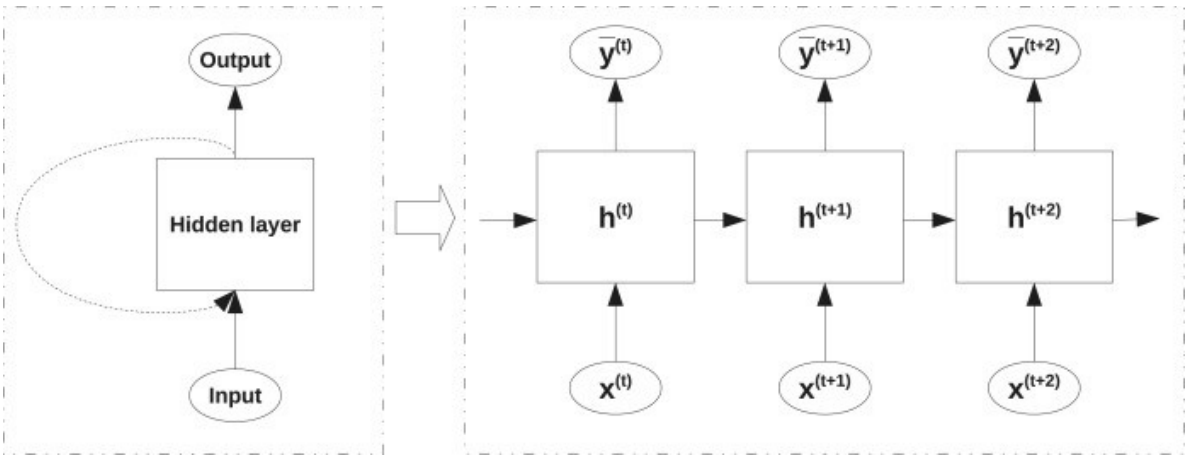

Figure 2.1: Training Model

I have combined several technologies together to make this a success. We have implemented several technologies such as deep learning, recurrent neural networks and used frameworks like IBM Watson. Deep learning is an empirical science, and the quality of a group's infrastructure is a multiplier on progress. Fortunately, today's open-source ecosystem makes it possible for anyone to build great deep learning infrastructure. A typical deep learning advance starts out as an idea, which you test on a small problem. At this stage, you want to run many ad-hoc experiments quickly. Ideally, you can just SSH into a machine, run a script in screen, and get a result in less than an hour. You need to inspect your models from many angles to gain intuition for what they're actually learning.

2.1.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are popular models that have shown great promise in many NLP tasks. The idea behind RNNs is to make use of sequential information. In a traditional neural network, we assume that all inputs (and outputs) are independent of each other. But

for many tasks that's a very bad idea. If you want to predict the next word in a sentence you better know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps.

2.1.2 Natural Language Processing

Natural language processing (NLP) is the ability of a computer program to understand human language as it is spoken. NLP is a component of artificial intelligence (AI). The development of NLP applications is challenging because computers traditionally require humans to "speak" to them in a programming language that is precise, unambiguous and highly structured, or through a limited number of clearly enunciated voice commands. Human speech, however, is not always precise – it is often ambiguous and the linguistic structure can depend on many complex variables, including slang, regional dialects and social context.

2.1.3 Optimisation and Training Data

For optimisation we have used Adam Optimiser instead of gradient descent to increase the performance of the model. At first, our model was unable to store some results in memory affecting the overall accuracy in case of long sentences in our model. But we rectified this issue by applying LSTM "Long Short-Term Memory". Once the model shows sufficient promise, you'll scale it up to larger datasets and more GPUs. This requires long jobs that consume many cycles and last for multiple days. You'll need careful experiment management, and to be extremely thoughtful about your chosen range of hyper parameters.

The training data consists of the following artifacts:

- Intents: Goals that you anticipate your users will have when they interact with the service. Define one intent for each goal that can be identified in a user's input. For example, you might define an intent named store hours that answers questions about store hours. For each intent, you add sample utterances that reflect the input customers might use to ask for the information they need, such as, What time do you open?

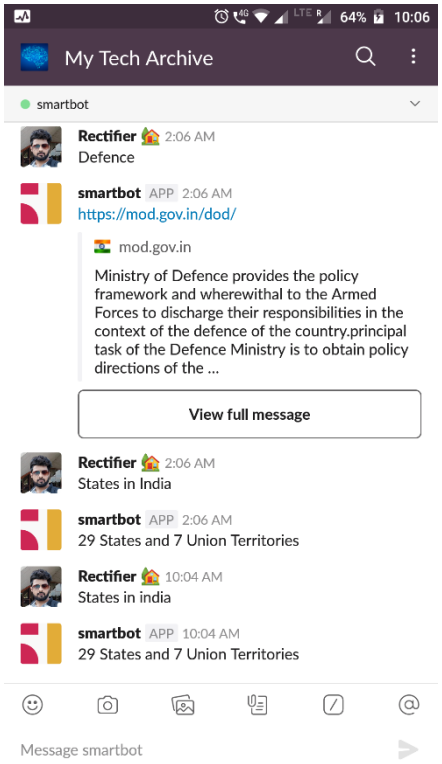

Figure 2.2: Screenshot of working UI

- Entities: An entity represents a term or object that provides context for an intent. For example, an entity might be a city name that helps your dialog to distinguish which store the user wants to know store hours for.
  As you add training data, a natural language classifier is automatically added to the skill, and is trained to understand the types of requests that you have indicated the service should listen for and respond to.

- Dialog: Use the dialog tool to build a dialog flow that incorporates your intents and entities. The dialog flow is represented graphically in the tool as a tree. You can add a branch to process each of the intents that you want the service to handle. You can then add branch nodes that handle the many possible permutations of a request based on other factors, such as the entities found in the user input or information that is passed to the service from an external service.

2.2 IBM Watson

IBM Watson Conversation service, you can create an application that understands natural-language input and uses machine learning to respond to users in a way that simulates a conversation between humans.Chatbots understand what we say by passing our text through an NLP engine, such as IBM Watson, which uses its gargantuan database to deduce the meaning of a sentence. The point of NLP is not to interpret sentences word-for-word but to extract the

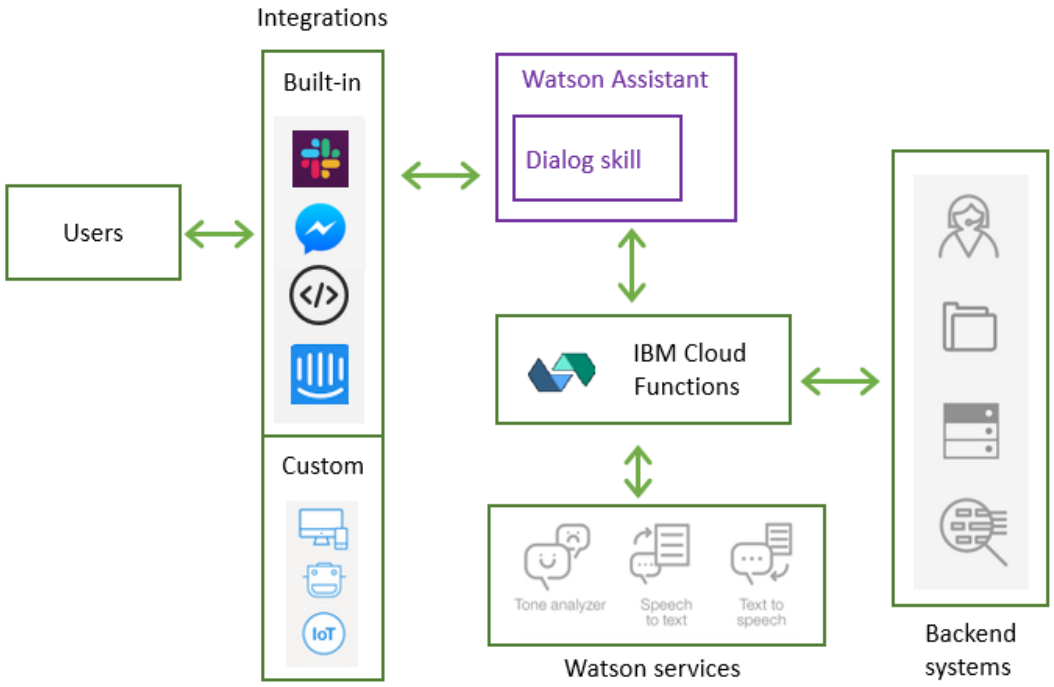

Figure 2.3: Architecture of IBM Watson

intent behind the message. Intents are like metadata that define your services. Watson Conversation uses concepts like intents, entities, and dialog to help you craft powerful conversational experiences. The above diagram shows the overall architecture.

- Users interact with the assistant through one or more of these integration points:
  1. A chat bot that you publish directly to an existing social media messaging platform, such as Slack or Facebook Messenger.
  2. A simple chat bot user interface that is hosted by IBM Cloud.
  3. Custom application that you develop, such as a mobile app or a robot with a voice interface.
- The assistant receives user input and routes it to the dialog skill.
- The dialog skill interprets the user input further, then directs the flow of the conversation and gathers any information that it needs to respond or perform a transaction on the user's behalf.

2.2.1 Deep Parsing in IBM Watson

Two deep parsing components, an English Slot Grammar (ESG) parser and a predicate-argument structure (PAS) builder, provide core linguistic analyses of both the questions and the text content used by IBM Watson to find and hypothesize answers. Specifically, these components are

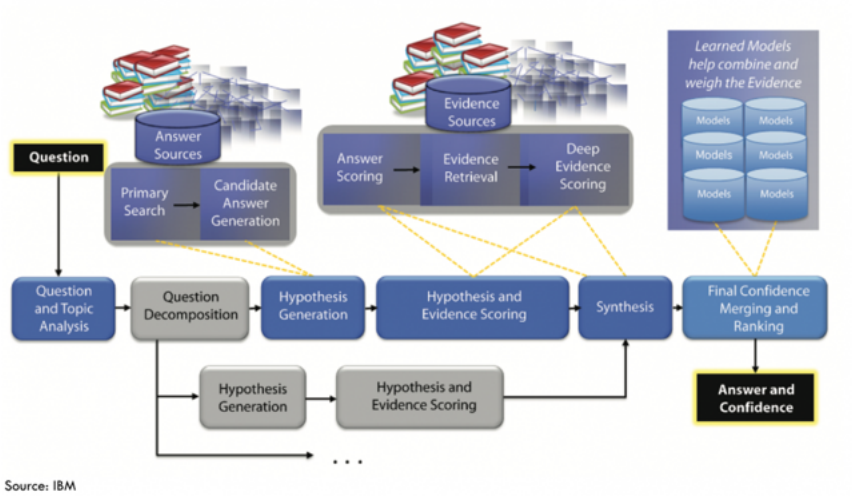

Figure 2.4: Parsing of data

fundamental in question analysis, candidate generation, and analysis of passage evidence. As part of the Watson project, ESG was enhanced, and its performance on Jeopardy! questions and on established reference data was improved. PAS was built on top of ESG to support higher-level analytics.

SG parsing The SG parsing system is divided into a large language-universal shell and language-specific grammars for English, German, French, Spanish, Italian, and Portuguese. Some of the SG features described in this section are in the shell, and some are specific to English syntactic analysis.

Predicate-argument structure The PAS builder provides simplification and abstraction of the ESG parse that removes some details. By design, the semantic distinctions that are removed in the PAS are ones that are subtle and not essential to the coarse-grained distinctions we make in many Watson components that use the parses. Without these distinctions, those components can be more flexible and require less knowledge. For example, passive and active forms of the same assertion result in the same PAS. The ESG parse does show the active-form (logical) arguments of a passive verb, except that a logical subject is a "by"-PP.

2.3 Workflow

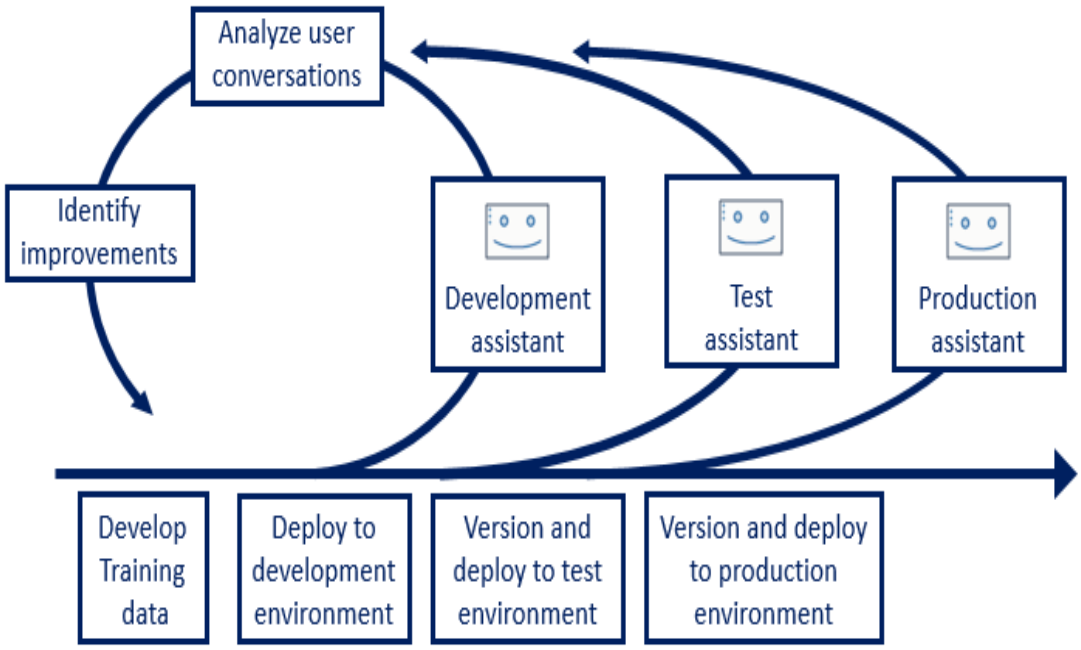

Figure 2.5: Workflow of IBM Watson

The typical workflow for an assistant project includes the following steps:

1. Define a narrow set of key customer needs that you want the assistant to address on your behalf, including any business processes that it can initiate or complete for your customers.

2. Create intents that represent the customer needs you identified in the previous step. For example, intents such as About_company or #Place_order.

3. Build a dialog that detects the defined intents and addresses them, either with simple responses or with a dialog flow that collects more information first.

4. Define any entities that are needed to more clearly understand the user's meaning. Mine existing intent user examples for common entity value mentions. Using annotations to define entities captures not only the text of the entity value, but the context in which the entity value is typically used in a sentence.

5. Test each function that you add to the assistant in the "Try it" pane, incrementally, as you go.

6. When you have a working assistant that can successfully handle key tasks, add an integration that deploys the assistant to a development environment. Test the deployed assistant and make refinements.

7. After you build an effective assistant, take a snapshot of the dialog skill and save it as a version. Saving a version when you reach a development milestone gives you something you can go back to if subsequent changes you make to the skill decrease its effectiveness.

8. Deploy the version of the assistant into a test environment, and test it. If you use the web-hosted chat widget, you can share the URL with others to get their help with testing.

9. Use metrics from the Improve tab to find areas for improvement, and make adjustments. If you need to test alternative approaches to addressing an issue, create a version for each solution, so you can deploy and test each one independently, and compare the results.

10. When you are happy with the performance of your assistant, deploy the best version of the assistant into a production environment.

11. Monitor the logs from conversations that users have with the deployed assistant.

CHAPTER 3

FEATURES

3.1 Privacy and Requirements

One of the main advantages that our bot has is that is availability, it can be accessed from anywhere, at any time using slack in which any one can join the bot channel and have a conversation according to their needs. Privacy is not a concern over here because many users can have a conversation with the bot at the same time in their own tabs. Other users will not be able to access data of other users so there not much of a privacy concern. But a user must have following in order to access our bot:

- Internet Connection
- PC/MAC/Smartphone

3.2 Backing Up

Backup and restore your data by exporting, and then importing the data. You can export the following data from a Watson Assistant service instance:

- Dialog skill training data (intents and entities)
- Dialog skill dialog

You cannot export the following data:

- Assistant, including any configured integrations

3.3 Advantages

My implemented bot is first of a kind, it replies more accurately as compared to other bots. This bot is for students, parents and teachers who are facing hard time either in studies or handling other personal issues. It also acts as a mentor for students of category which are unable to take important decision of their life. Our bot's accuracy will get better as we feed more data to the bot. We are hoping that someday we will be able to pass the Turing test for this bot.

3.4 Flow Chart

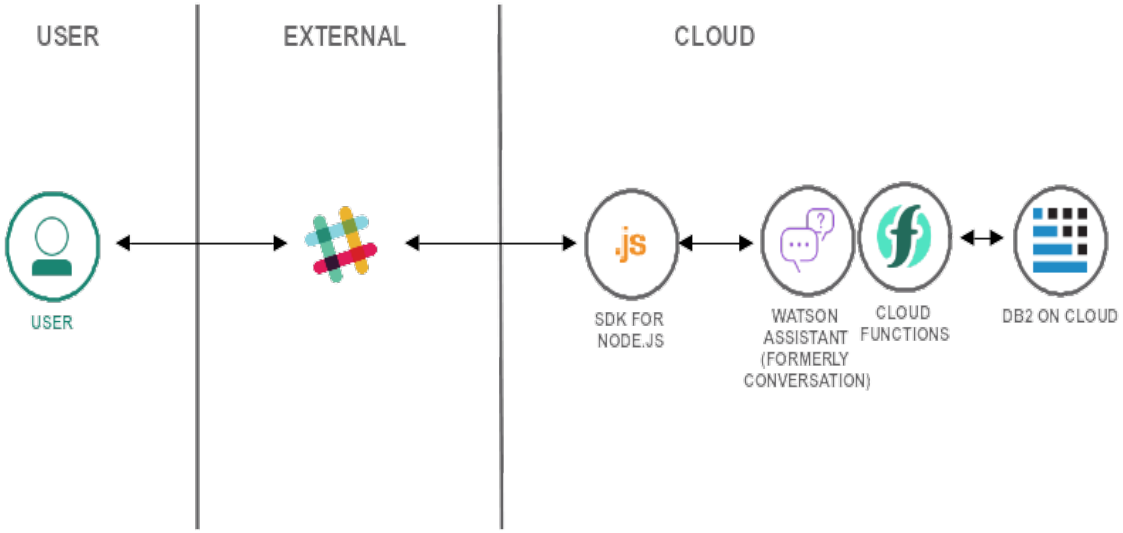

Figure 3.1: Information exchange using Slack API

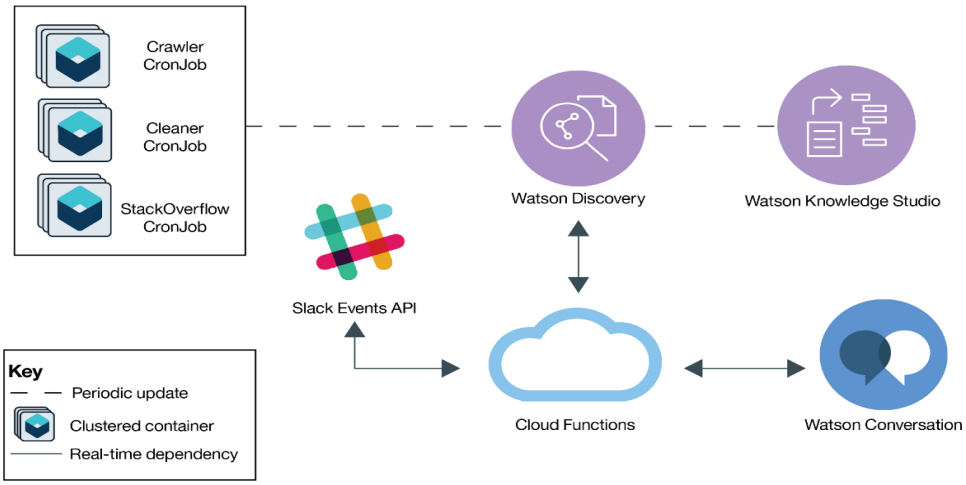

Figure 3.2: Information exchange in Watson

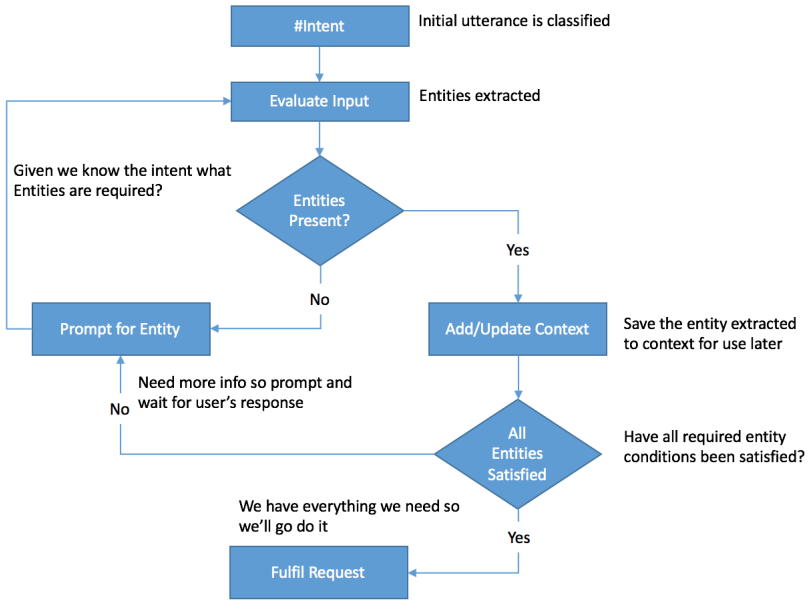

Figure 3.3: Watson Conversation

3.5 Data Flow Diagram

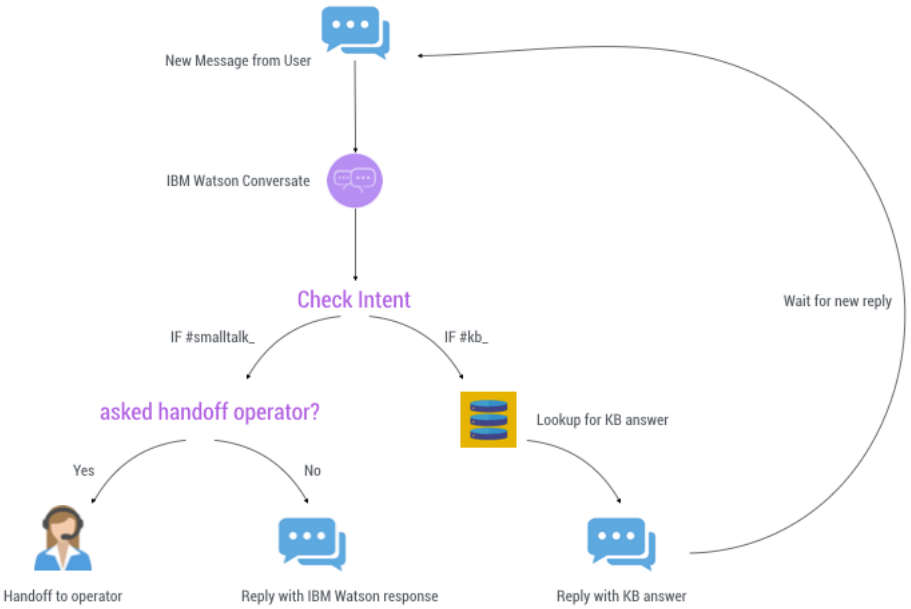

Figure 3.4: Data Flow in Watson

CHAPTER 4

TESTING

Three main testing methods:

4.1 80% / 20% Split

The 80% / 20% split is the most basic approach and probably the most common. Users hold back 20% of their GT (Ground Truth, or all the data points for the chatbot) rather than training with the entire GT. Then, after major changes are made to their development chatbot, they use the 20% GT to test the accuracy and ensure nothing has regressed since the last update. Accuracy of a chatbot can be defined as the percentage of utterances that had the correct intent returned. Imagine we have a chatbot that talks about dogs (since dogs *are better* than cats). I have this chatbot in production already and it has 10 intents. I'm about to release v2 of the DogBot, which changed my GT a bit and added two more intents. I promote my development version to my staging environment, and run my script that sends in each utterance from my held back 20% GT matching the returned intent to the expected intent. If the overall accuracy is the same or greater than my production bot from the last time I ran the tests, then my new changes have the green light to be promoted to production. Why use 80% / 20%? It's simple and easy to implement. However, it can be less accurate than the other two options. Also, you will end up not using your full GT in your production bot so the bot isn't biased during testing.

4.2 K-Fold Cross Validation

The second method, K-Fold Cross Validation, divides the training set (GT) into K number of parts (folds), then uses one fold at a time as the testing fold and the rest of the data as the training data. The most common test is 5-fold, but you can choose whatever number you'd like. This means the training data is split into five folds. The bot is trained with four of the folds, and the fifth fold is used to test (just like the 20% GT in the above example). Repeat

this so each fold has a turn as the testing fold. Afterwards, average the overall accuracies of the folds together to get the accuracy of the chatbot. K-Fold is better than 80% / 20% because it gives you five times the number of tests, making it more robust. However, this is a bit more complicated to implement since you have to create each fold.

4.3 Monte Carlo Cross Validation

Monte Carlo is similar to K-Fold except the data sets are determined randomly. Shuffle the data randomly, pick the first 80% as training and designate the rest as testing data. Or, another way is to randomly split the data into an expected ratio. Then, you have one dataset for testing, and you repeat the shuffling multiple times so you can get as many splits as you want. Generally, the data sets are shuffled five times giving five accuracies to average together. This provides the most random data sets of the three options, but the different splits could contain overlapping data since it is randomly selected.

4.4 Implementation of Cross Validation

We will be using Watson Conversation to build our bot and test its accuracy. To demonstrate, we will be using the demo workspace that comes with Watson Conversation: The Car Dashboard. There are 27 intents in this workspace. The first step is to get our data in CSV format. Log into your workspace and go to the Intents page. Click the checkbox next to the Intents title to select all of your intents and export (shown below)

That's all we need to do for our data! The script will handle the rest once the data is in this format. For this script, we are mixing Monte Carlo with a normal K-fold test. The data is shuffled before the folds are created (Monte Carlo) but the folds are still unique (Normal K-fold). You can find the code for the script here: https://github.com/ammardodin/conversation-cross-validation Clone the repo to your computer and open with an editor. Open conversation_cv.py and add your Conversation credentials on line 66. Make sure you have enough space in your instance before proceeding. This will create a workspace for every fold you create. Next, we need to download our dependencies. we use virtualenv for this, meaning the first step is to create the environment with virtualenv ENV then activate it with source ENV/bin/activate. Download the

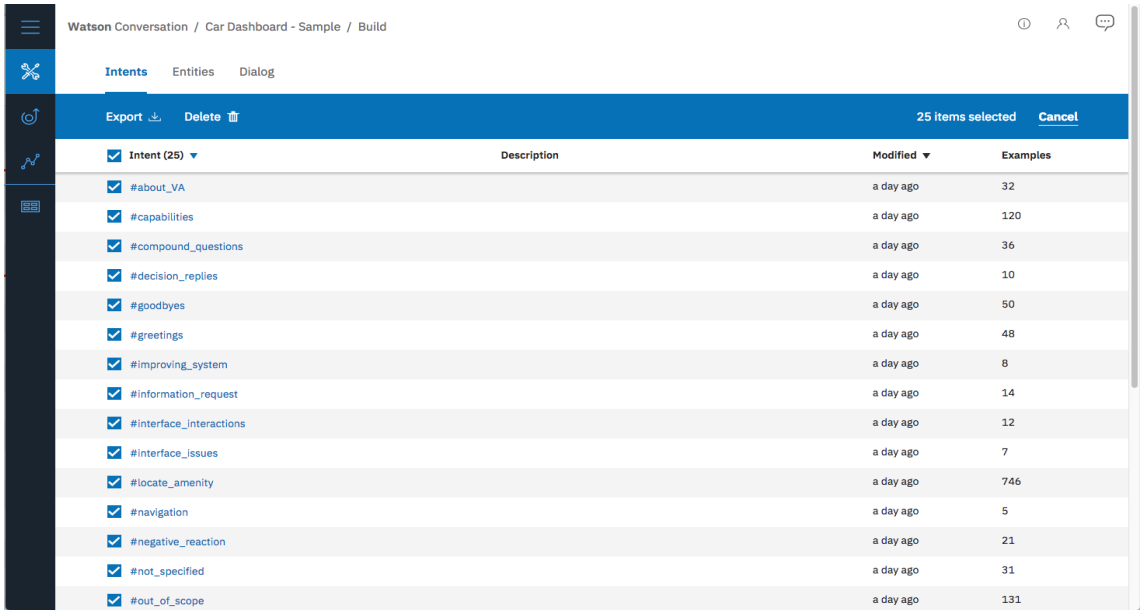

Figure 4.1: Screenshot of intents

dependencies with pip install -r requirements.txt. Make sure to move the exported CSV to the same folder as the repo you clone. Then, run this command: python kfold.py –data data.csv –folds 5. The data flag will allow you to specify what dataset you are uploading, and the folds flag will allow you to choose how many folds you want to split your data into. A good default here is 5. The script will print to your terminal the accuracy of a particular fold while the others are used for training, as well as the averaged accuracy. Here's the result of my test using the Car Dashboard workspace and 5 folds:

Accuracy: 0.8929

Accuracy: 0.8889

Accuracy: 0.9147

Accuracy: 0.9167

Accuracy: 0.9127

Average accuracy: 0.9052

Woo! Well done.

4.5 Conclusion of Testing

Using either cross validation method will provide a more accurate measure of how well your chatbot is performing. The major difference between Monte Carlo Cross Validation and K-Fold is you don't have to split it into 5 fold upfront and combine them in combinations. The random shuffle is in theory less work, and you pick the top N% as train data and the rest as test. Every time a decent sized change is made to your intents, run this script in your staging environment to double check your accuracy before proceeding to production. If you want to add on to the script, it'd be helpful for it to list what intents were missed the most so you can easily debug.

CHAPTER 5

CONCLUSION

This project presents my idea of a bot that is capable to helping many individuals in different ways. It describes the need of such a model and how it would help many people in the society. Basic concepts about Recurrent Neural Networks and Natural Language Processing are discussed. Optimization Techniques are also discussed along with training data. The core of the project i.e. *IBM Watson* have been explained in an elaborate manner which constitutes of Architecture, Deep Parsing and Workflow. Furthermore, several other advantages of my model are discussed along with Privacy and Backup policies. There are several charts and diagrams which explains the flow of information and the exchange of data between the inner components of IBM Watson. Moreover, testing of the whole model is discussed thoroughly and several methods of validation are discussed. Accuracy of the model is also mentioned in the testing section and methods are also explained to perform testing on similar models.

CHAPTER 6

FUTURE ENHANCEMENT

It is an open question why my model recovers the concept of personal assistant in such a precise, disentangled, interpretable, and manipulable way. I am further planning to improve the bot by integrating sophisticated technologies into my bot. It will be given more training time on a large data set enhancing its overall accuracy. In this bot we can also implement web scraping from forums to provide with more relevant answers to the queries of the user. Speech recognition is another aspect I am considering for this bot to make it more interactive to the user and more reliable. As we all know, computer vision algorithms are getting stronger day by day we are going to implement these algorithms into our bot to make our bot more advanced because by this technology our bot will be able to recognize the current mood of the user just by using device's camera on which our user is working on. This bot will not be limited to some languages; we are planning to add more languages so that it impacts more people who are confused but are willing to work hard.

CHAPTER 7

REFERENCES

- Deep Learning by Ian Goodfellow
- IBM Watson Assistant User Guide
- Deep Parsing in IBM Watson by M.C. McCord, J.W. Murdock, B.K. Boguraev
- Kalchbrenner, N., Grefenstette, E. and Blunsom, P., 2014. A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188
- Blei, David M, Ng, Andrew Y, and Jordan, Michael I. Latent dirichlet allocation. Journal of machine Learning research, 3(Jan):993-1022, 2003
- Chen, Danqi and Manning, Christopher D. A fast and accurate dependency parser using neural networks. In EMNLP, pp. 740-750, 2014
- Kim, Yoon. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882, 2014
- B. Han, P. Cook, and T. Baldwin, "Lexical Normalization for Social Media Text," ACM Trans. Intelligent Systems and Technology, vol. 4, no. 1, 2013
- R. Fernandez, J. Ginzburg, and S. Lappin, "Classifying Non-Sentential Utterances in Dialogue: A Machine Learning Approach," Computational Linguistics, vol. 33, no. 3, 2007, pp. 397-427
- K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. IJCAI-99 Workshop on Machine Learning for Information Filtering, pages 61-67, 1999
- K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. Machine Learning, 39(2/3):103-134, 2000
- G. Salton and M. McGill, editors. Introduction to Modern Information Retrieval. McGraw-Hill, 1983