Unibot

A multipurpose chatbot generalised for end-user applications

Interns

Sonam Shenoy	PES1201801282
Tejas Srinivasan	PES1201800110
Taran Singhania	PES1201800333

Mentors

Ishaan Lagwankar	PES1201700150
Tejvi M	PES1201700119

Microsoft Innovation Lab
PES University
Summer Internship
June - July 2019

Abstract

Chatbots are starting to have an increasing presence in our daily life. Almost every "customer support" agent is a chatbot. However, most chatbots are usually tailored for a specific task and use retrieval based systems. To solve this shortcoming, we proposed a hybrid model that uses both generative and retrieval based systems to give meaningful responses. We have also introduced generalization, by means of which users can feed their datasets and the model which trains on the cloud, should be able to create a chatbot for their needs. Given new data, using the dataset extension feature, the chatbot can be personalised to meet the user's expectations. Based on the context of the conversation and the sentiment of the user, the decision system of the model decides an appropriate time to respond to the user's query appropriately.

Contents

Chapter 1

Introduction

Recurrent Neural Networks (RNNs), which are special kinds of neural networks particularly adept at dealing with sequential data (like text sequences) are an integral part of our chatbot. Bidirectional RNNs that use Encoder-Decoder Netowrks can be used to detect the context of a sentence from different positions relative to a target word. This makes the bot more accurate in its reply. Natural Language Processing and its sub domain Natural Language Understanding (NLU) are used for tasks such as sentiment analysis and classification of conversations into buckets. Web Design and Cloud Interfacing is used to make it easy for the user to interact with the bot and a have a hassle free conversational experience.

The bot solves the problem of creating separate specialized bots for different tasks by providing a 'one for all platform'. Further it addresses the problem of accuracy and contextual richness of responses over a period of time that is essential for a great personalized experience.

Since our bot is generalized , it can have applications for various use cases. The two main use cases would be personalized assistants for users as well as delivery agents and event schedulers.

1.1 Problem Statement

To create a generalized chatbot which can intelligently respond to conversation of various genres and contexts by continuously improving on human conversation by using a follow-up mechanism using a Seq2seq and an Attention model.

For a lucid reading experience, the rest of the report has been divided into several sections.
1) Literature Survey/Related Work
2) Weekly Progress Report
3) Block diagram/Flow chart
4) Hardware and Software Requirements
5) Web Design and Cloud Interface

Chapter 2

Literature Survey/Related Work

The literature survey that was carried out in the first week of the internship, involved exploration of the fields of Natural Language Processing (NLP) using various NLP libraries such as SpaCy and NLTK. Named Entity Recognition (NER) and Cosine similarity were done on test sentences. A quick overlook of the required architectures for Recurrent Neural Networks, Word2Vec models, LSTMs and GRU's was done as a part of the Deep Learning Specialization on Coursera. Machine Learning libraries such as Tensorflow, PyTorch and Keras were explored and a basic Natural Language Generation Model was developed on basic training data. A brief exploration into Reinforcement Learning Algorithms for text based processing was also done, but the existing work was beyond the scope of the project and was subsequently dropped.

Further, blog posts, course lectures and papers were studied in detail and numerous observations were made-

1) There are many models which can perform several tasks well. However, it was observed that none of them were truly generalised and adaptable to the dataset provided as well as the proposed model. Furthermore, the interface provided for a simple drag and drop of the dataset offers the user convenience and does not require them to install any libraries or modules which is a hassle, undoubtedly.

2) Chatbots such as Juji (retrieval based) , Neuralconvo (purely AI based) etc. did not use a hybrid model as is proposed and thus did not achieve results which were obtained through a trade off between the two models.

Furthermore, none of the bots had an option of dataset extension that would make the bot more personalised over time and sensitive to the user's needs. This was a novel feature that set our chatbot apart from the existing alternatives.

3) Many existing chatbots (both retrieval based and generative) were explored but none of them were found to have a properly implemented follow up mechanism that is so essential to have long term relations and meaningful conversations with the user. The follow up mechanism can have many applications as mentioned in the appropriate section.

All the resources used for the Literature survey have been included in the References section of the report.

2.1 Generalization

Most of the chatbots in the current market are tailored to perform a specific task. Although these bots perform well on the task at hand they usually fail at other simple tasks like maintaining simple conversation for long periods of time and contextual responses.Most bots are made for specific tasks, our generalization model basically aims to adapt to datasets of different contexts which can just be uploaded onto the cloud. These files automatically get converted to text files which in turn are converted into metadata for processing and training. Thus, all the user has to do is upload their dataset in the given format as a CSV or JSON. The model then takes care of the rest of the work as mentioned above.

2.2 Dataset Extension and Retrieval

As a boost to the replies the bot can give, it is now made possible for the user to add custom inputs as per their preferences if they aren't satisfied with the ones the bot already gives. These custom input and preferences, are stored in a database. The bot considers them the next time the user converses with it and enters the same or a similar query by retrieving from this database. Over a series of conversations with the same user or with multiple users the bot learns the pattern of conversation and gives better, context-rich responses using a NLG (Natural Language Generation) model.

2.3 Follow Up

Follow up, a key feature of the bot, is one in which, upon terminating a conversation with the user, it stores the user's input in a database based on his sentiment. Using the NLG model the bot interprets the context of the conversation and extracts the aspects of the conversation. A threshold and polarity for user sentiment is set and accordingly a decision system decides the time delay and the response (depending on the context) and thus appropriately generates a follow up query to the user. This follow-up mechanism essentially reinforces the personalization aspect in bot-human interactions by asking and can serve as a reminder system to provide system updates after a defined interval of time.

Chapter 3

Concepts Used

3.1 Word Embeddings

Word Embedding is a technique for learning dense representation of words in a low dimensional vector space. Each word can be seen as a point in this space, represented by a fixed length vector. Semantic relations between words are captured by this technique.
Word Embedding is typically done in the first layer of the network : Embedding layer, that maps a word (index to word in vocabulary) from vocabulary to a dense vector of given size. In the seq2seq model, the weights of the embedding layer are jointly trained with the other parameters of the model.

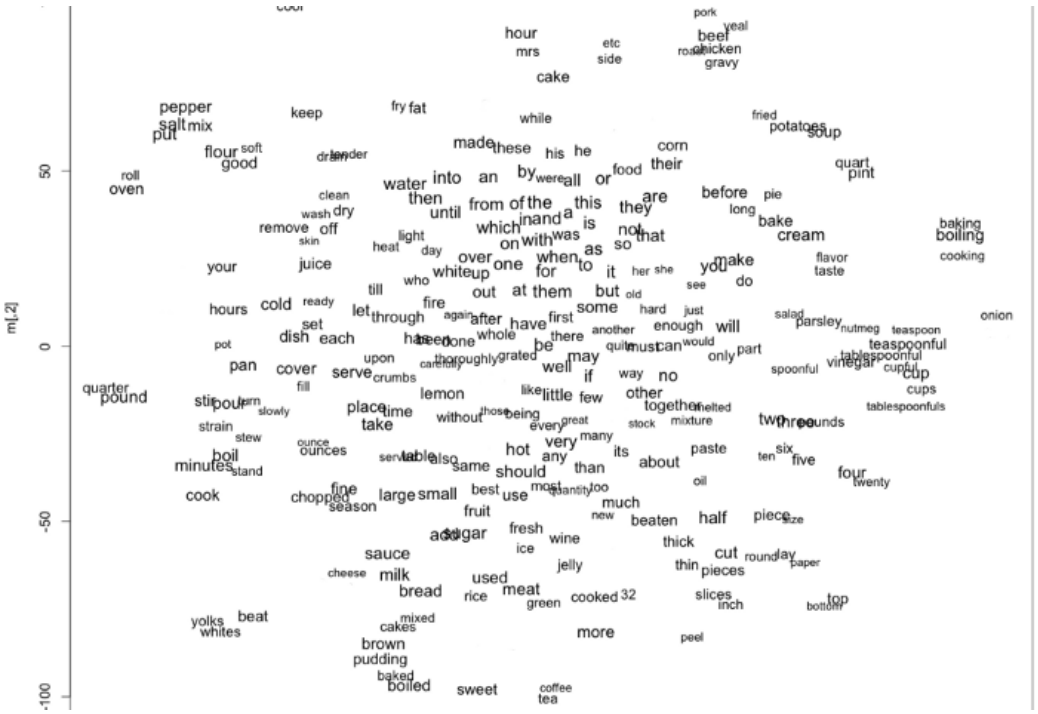

Figure 3.1: Word Embeddings

3.2 Recurrent Neural Networks(RNNs)

Recurrent Neural Networks are a class of artificial neural networks where the output of the previous step is fed to the input of the current step. Traditionally artificial neural networks are independent of output of each other. But in cases where the word in a sequence is dependent on the previous words, previous input is required. RNNs solve the issue by adding a "Hidden Layer" which basically acts as a memory to remember the previous context. They provide the same weights and biases to all the layer, thus reducing the complexity of the increasing parameters. However a disadvantage of RNN is the vanishing gradient problem. For longer sequences it becomes hard for the RNN to tune the hyperparameters of the earlier layers of the network and thus the previous context is forgotten.

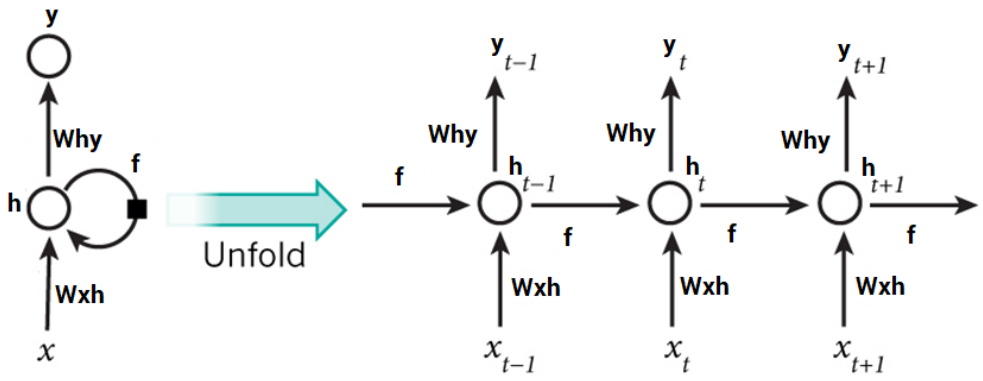

Figure 3.2: Recurrent Neural Networks

3.3 Long Short-Term Memory Networks (LSTMs)

Long Short-Term Memory Networks are a special kind of Recurrent Neural Network that are used for handling long-term dependencies. This is done because LSTMs have the ability to selectively remove or add information to the cell state using regulated structures called gates.

Bidirectional LSTMs (BiLSTMs) are a variant of LSTMs designed to implement the learning algorithm on the original data from beginning to end and once from the end to the beginning simultaneously which helps in a much better understanding the context of the conversation.

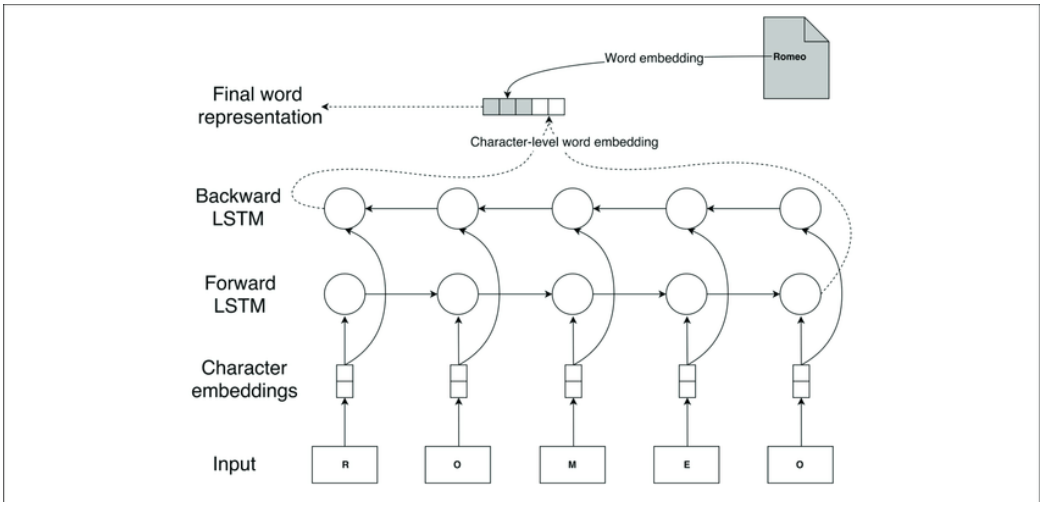

Figure 3.3: Representation of Long Short-Term Memory Networks

3.4 Seq2seq model

Sequence To Sequence model consists of two RNNs (Recurrent Neural Network) : An Encoder and a Decoder. The encoder takes a sequence(sentence) as input and processes one symbol(word) at each timestep. Its objective is to convert a sequence of symbols into a fixed size feature vector that encodes only the important information in the sequence while losing the unnecessary information.
Each hidden state influences the next hidden state and the final hidden state can be seen as the summary of the sequence. This state is called the context or thought vector, as it represents the intention of the sequence. From the context, the decoder generates another sequence, one symbol(word) at a time. Here, at each time step, the decoder is influenced by the context and the previously generated symbols.

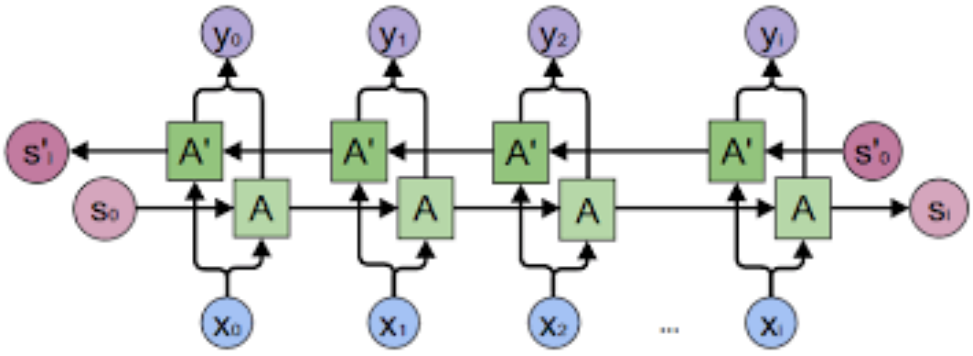

Figure 3.4: The Seq2seq network using Bidirectional RNNs

3.5 Attention model

One of the limitations of seq2seq framework is that the entire information in the input sentence should be encoded into a fixed length vector, context. As the length of the sequence gets larger, we start losing considerable amount of information. This is why the basic seq2seq model doesn't work well in decoding large sequences. The attention mechanism, introduced in this paper, Neural Machine Translation by Jointly Learning to Align and Translate, allows the decoder to selectively look at the input sequence while decoding. This takes the pressure off the encoder to encode every useful information from the input.

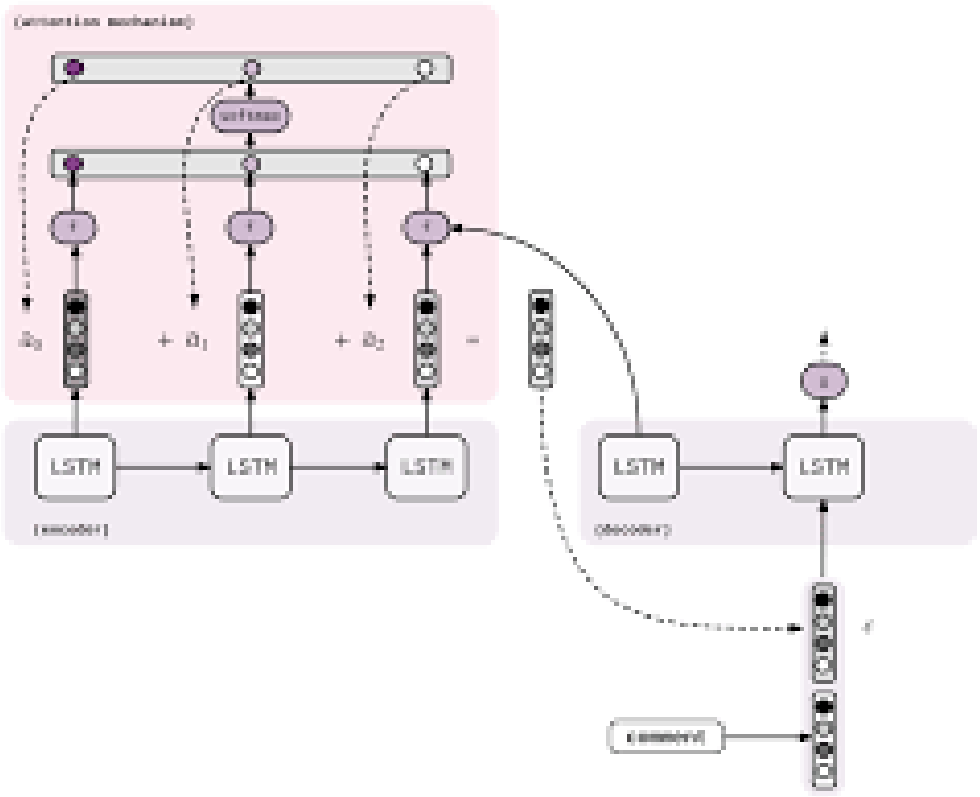

Figure 3.5: Attention mechanism

3.6 Aspect Extraction

An end to end neural dialogue model called Attention Based Aspect Extraction used unsupervised methods to extract the sentiment and keywords from the user's input which usually has a multidimensional aspect overview.

- Sentiment Extraction Sentiment analysis is a branch of text classification where the sentiment terms (feature,opinion etc) that have aspects of polarity and subjectivity are extracted and their semantic opinions are determined. The model uses implicit ABSE (Aspect Based Sentiment Analysis) to extract user's sentiment and then deliver that sentiment to the decision system that uses a Natural Language Generation Model.

- Keyword Extraction The model also takes care of factual questions and responses by using implicit keyword extraction using the neural model to extract the subject, context and other keywords from the user query and passes it to the MediaWiki API for searching through Wikipedia for suitable keyword based, short responses.

3.7 Multi threading

A thread is basically a separate flow of execution. Essentially it means that different independent tasks run concurrently in Python. Multiple threads within the same data space can communicate with each other and are thus less memory and time intensive. These threads also help in modularizing the tasks and help in easier debugging of errors. These threads can be implemented using the Threading module in Python using the ThreadPoolExecutor function.

Chapter 4

Weekly Progress Report

4.1 Week 1 (3rd June to 7th June)

The literature review consisted of exploring the field of Natural Language Processing (NLP) using the NLP library Spacy and learning about the architecture of RNN and LSTM. Named Entitty Recognition (NER) , Part-of-speech tagging (POS) and Dependency Parsing were done using the Spacy Library. The theory involved was studied in detail and papers published and related work done in the field was studied as well. Furthermore the previous projects done on the same domain were used as a reference for practical advice.

Different interfaces and modules were introduced such as Textblob and nltk (For sentiment analysis) , tensorflow , keras and many more.

4.2 Week 2 (10th June to 14th June)

Week 2 started off with implementation of a basic follow-up model. The user's input was returned from a database (which was made using SQLite3) and a sleep timer of different time intervals were given. Sentiment Analysis on commonly encountered sentences in conversation was performed using TextBlob and the sentences with polarities opposite to those with negative sentiments were generated.

4.3 Week 3 (17th June to 21st June)

During Week 3 , a Natural Language Generation Model (NLG) was built using Spacy's Matcher module and the model was trained using Pytorch , a deep learning framework. The model was trained on the Questions taken from the Stanford Question Answering Dataset and questions were generated based on a list of keywords taken from the user's input. The bot was made interactive (a proper conversational flow between the bot and the user was established) and a more advanced version of the follow-up mechanism was introduced based on sentiment of the user. A reinforcement learning approach was also considered and an additional literature survey was done on that topic.

4.4 Week 4 (24th June to 28th June)

The fourth week involved developing the User Interface (UI) using HTML, CSS, Bootstrap and Django. The chatbot model was trained on the Twitter dataset along many other datasets (Stanford Question Answering Dataset - SQuaD, Cornell Corpus dataset) and deployed on the cloud using Heroku , so as to make it easier for the user (user need not train the model on their system) to interact with the chatbot.

4.5 Week 5 (1st July to 5th July)

The fifth week was dedicated to improving the follow up mechanism of the chatbot based on sentiment, frequency of questions etc. and the concept was broadened to include reminders too. The retrieval model was built to retrieve previous inputs and conversations from the user's chat database and the dataset extension feature was built to extend the model's trainset based on the user's input.

The idea of using Transfer learning to improve the generalization in the model was dropped after seeing that padding the sentences would only lead to loss in contextual information and would increase processing time threefold.

4.6 Week 6 (8th July to 12th July)

The sixth week involved exploration of IBM's Arria Studio , an NLG platform that describes structured data in human understandable language, Furthermore , the MediaWiki (for factual queries) and Dialogflow API's were integrated to improve model performance. The User Interface was also improved upon by adding animations and dynamic elements using Javascript.

4.7 Week 7 (15th July to 19th July)

The seventh week included training the model and further optimising the hyperparameters to improve the bots response. The model was trained on a NVIDIA GT 730 for 2 days on each dataset to add on to the models accuracy.

4.8 Week 8 (22nd July to 26th July)

The UI was updated with the main features that include dataset extension and follow-up. Now, the user can customize with their own questions and answers if they aren't satisfied with the bots reply. Also follow-up was integrated with the website so that it asks or reminds the user about their status after a particular time depending on the seriousness of the situation. The website was thus made live after it was tested in the team.

Chapter 5

Block Diagram/Flow Chart

5.1 Encoder-Decoder Network

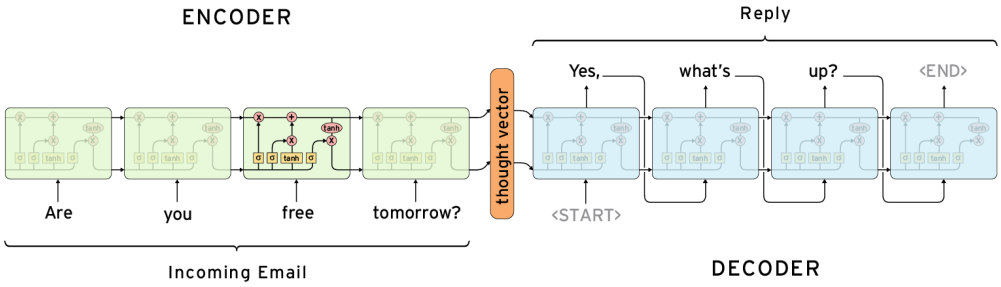

Figure 5.1: The Encoder-Decoder Network

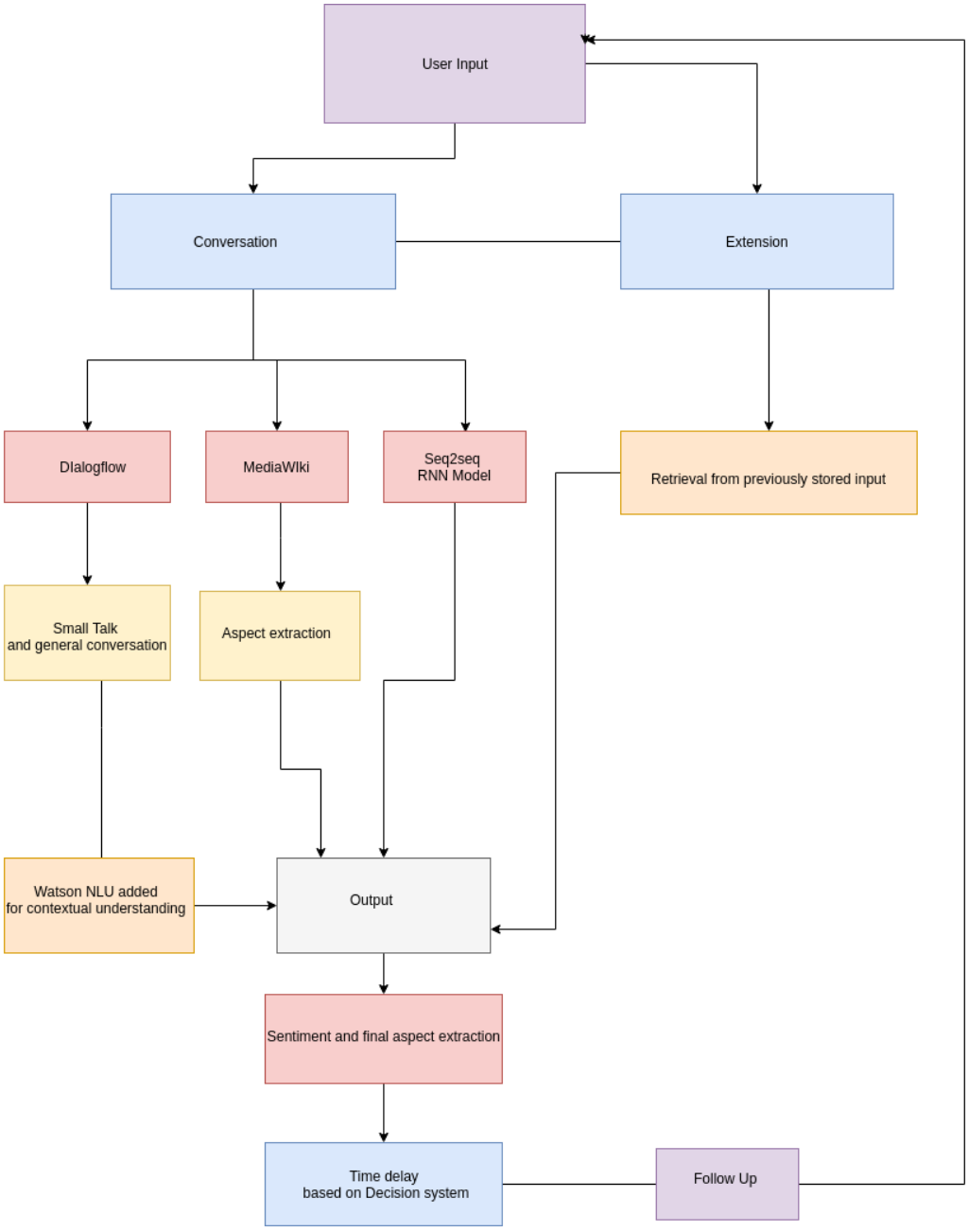

Figure 5.2: Conversational Flow and Follow Up

Chapter 6

Hardware and Software Requirements

To format code and run it locally, these libraries are required :-

Tensorflow 2.0
Tensorlayer
Spacy
Textblob
Numpy
Pandas
tqdm
Wikipedia
IBM Watson NLU
DialogFlow

There are no specific hardware requirements as such.

Chapter 7

Web Design and Cloud Interface

Link to the website can be found **here.**

To ease the way users can access and interact with the chatbot (unlike the less preferred terminal interaction), a user-friendly UI has been developed. Due to this feature, users don't have to download the code or install **any** software on their systems. All they have to do is go to the website and interact with it.

For the website, the Django Framework has been used for the back-end and Bootstrap for the front-end. The website has been deployed on the Cloud Platform - Heroku so as to enable customers anywhere to access it.

This website includes three features - one for dataset extension wherein the dataset should be present either as a CSV or JSON. The user is supposed to make sure that their dataset is present in the format shown in the website. Another feature "Start Conversation", the integral part of our project, is where the user interacts with the chatbot, and the follow-up taken care of. The last feature, i.e. to extend the dataset is also provided to the user so that he can add his own custom questions and answers.

To keep a track of the custom inputs and preferences of the users separately, an account has been created for each user due to which they have to login to play around with the website.

Each user's data is stored with the help of Firebase which makes it faster to accesss and keeps the information of the user secure. It also takes care of

the authentication process by adding a sign in feature which makes it ideal to integrate with Heroku.

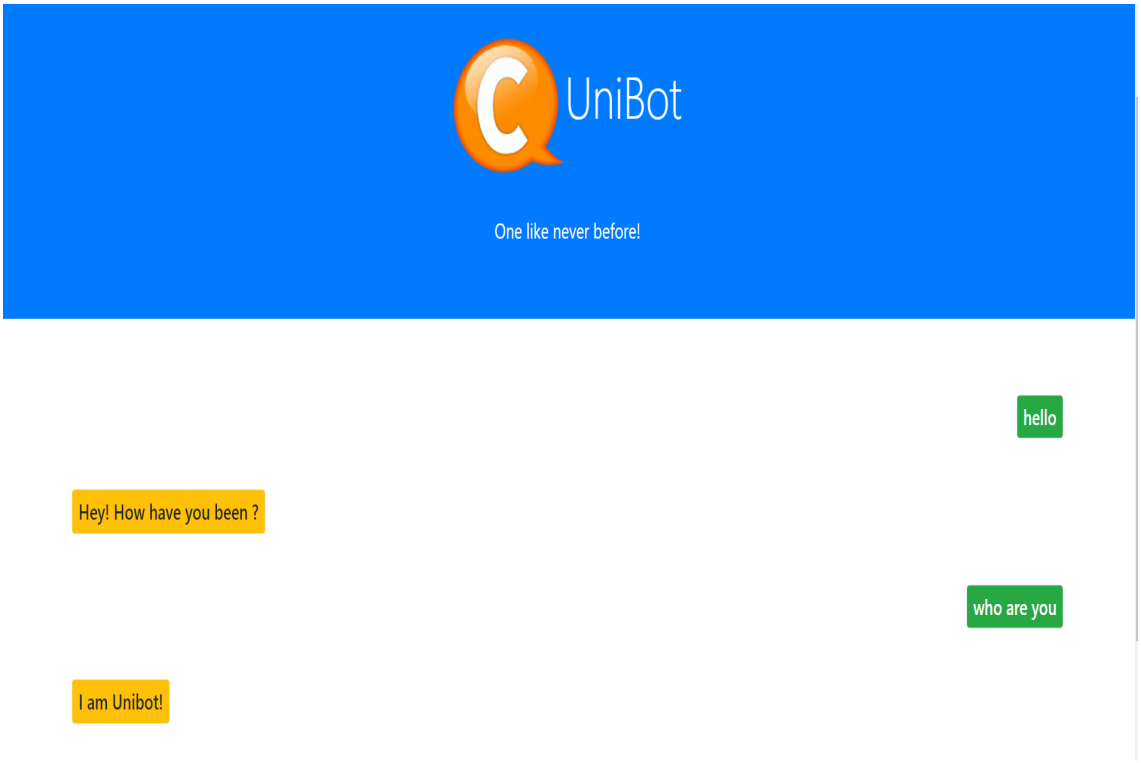

Figure 7.1: The Web Interface deployed on Heroku

Chapter 8

Results and Discussion

A generalized goal-oriented chatbot that follows up according to user preferences, after an appropriate time with a contextual response using a user friendly interface.

There were many challenges along the way:

Obtaining the required data for Data preprocessing.

Cleaning the dataset to make a format for generalization took a long time and proved to be tedious.

The time taken for training was longer than expected without which it was difficult to proceed with the next task

It took a long time to load the dataset in the website due to which multithreading had to be used.

Accuracy of response and context-rich conversations with users is still a challenge without the NLG model and a Reward system based on Reinforcement Learning.

A comparison of results obtained on two of the datasets (SQUaD for factual and (Cornell Corpus for conversational) is given below -

Criteria	SQuAD	Cornell Corpus	Twitter Dataset
Loss	0.97	0.76	0.89

Comparison of the three models

Chapter 9

Conclusions and Future Work

In conclusion, implementing a hybrid model (generative and retrieval based) leads to the best output. For the generative model, a Bidirectional RNN was found to yield the best output sequence in terms of context of the user's query. IBM's Watson NLU was found to be the best tool for sentiment analysis and aspect extraction. To add small talk and common phrases to the model, Google's dialogflow API proved to be the best tool.

The code for the model is uploaded on Github. The instructions to replicate the code have been provided in the Readme file.

The link for the same has been included in the references section.

There is much scope to improve the chatbot in the future. The main features of our model viz. Generalization, Dataset Extension can be improved much further given more user data of different kinds. Imitation Learning (a form of Reinforcement Learning) can be implemented to penalize or reward the bot based on the interest of the user in the conversation and the sentiment score of the conversation. This will also further help in personalizing the bot for the user.

A Natural Language Generation Model using Denoising Autoencoders can also be used to improve contextual responses and time-delay decision system for the follow-up mechanism.

The User Interface and Experience can also be improved much more, possibly through the development of an Android application to create a more accessible platform for the user to interact with the chatbot and notifications for follow up.

To improve the speed and efficiency of model training on the cloud, distributed learning algorithms such as Federated learning can be implemented.

Chapter 10

References

The references to the model and the API's are -

1. Practical Seq2seq: http://complx.me/2016-12-31-practical-seq2seq/
2. MediaWiki API Documentation: https://www.mediawiki.org/wiki/Documentation
3. DialogFlow Documentation: https://cloud.google.com/dialogflow/docs/
4. IBM Watson NLU: https://cloud.ibm.com/apidocs/natural-language-understanding
5. Link to Github Repository: https://github.com/tejvi-m/chatbot-MIL

Courses -

1. Spacy NLP course https://course.spacy.io/
2. Coursera Deep Learning Specialisation coursera.org/learn/nlp-sequence-models
3. Stanford NLP Course https://www.youtube.com/watch?v=OQQ-W 63UgQlist=PL3FW7Lu 3i5Jsnh1rnUwq TcylNr7EkRe6
4. David Silver Course on Reinforcement Learning https://www.youtube.com/watch?v=2pWv7G OYHWgPebj2MfCFzFObQ
5. Django Documentation https://docs.djangoproject.com/en/2.2/

6. Bootstrap Documentation https://getbootstrap.com/docs/4.1/getting-started/introduction/

7. Heroku Documentation https://devcenter.heroku.com/articles/deploying-python

8. Firebase Documentation https://firebase.google.com/docs

Tutorials -

1. https://keras.io/examples/lstm text generation/

2. https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/

3. https://www.tensorflow.org/tutorials/representation/word2v

4. https://firebase.google.com/docsec

Papers -

1. Contextual Neural Conversation https://arxiv.org/abs/1906.02738

2. End to End Task Completion https://arxiv.org/pdf/1703.01008.pdf

3. Short Text Conversation https://aclweb.org/anthology/D13-1096

4. Task Oriented Dialogue modeling https://arxiv.org/abs/1810.00278

5. Reinforcement Learning https://github.com/junhyukoh/deep-reinforcement-learning-papers

Other resources -

1. https://www.gutenberg.org/
2. https://github.com/shashank-bhatt-07/Natural-Language-Generation-using-LSTM-Keras/blob/master/Natural%20Language%20Generation%20using%20LSTM-Keras.ipynb