

# Project Report

---

## Title:

## **Building a Chatbot using Neural Networks and Natural Language Processing**

Derin Wilson

Roll no.18082

3<sup>rd</sup> year BS-MS, Physics

Indian Institute of Science Education and Research, Bhopal

## Introduction:

### **What is a Chatbot?**

A chatbot is an intelligent piece of software that is capable of communicating and performing actions similar to a human. Chatbots are used a lot in customer interaction, marketing on social network sites and instantly messaging the client. There are two basic types of chatbot models based on how they are built; Retrieval based and Generative based models.

We will be building a Retrieval based Chatbot

### **Retrieval Based Chatbots:**

The proposed system is a retrieval based system which particularly concentrate on a specific domain. A closed domain of Hospital query is taken for implementation. Retrieval based bots works by using the principle of directed flows or graphs. Basically such type of bots are trained to rank

the best response from a finite set of predefined responses. The responses here are entered manually by the developer, or based on a knowledge base of pre-existing information. The system is trained in such a way that there will be a set of questions that usually ask by the user multiple times in multiple ways and also set of corresponding responses

## About the project:

In this project we are going to build a chatbot using deep learning techniques. The chatbot will be trained on the dataset which contains categories (intents), pattern and responses. We use a neural network to classify which category the user's message belongs to and then we will give a random response from the list of responses.

We will be using Python, NLTK( Natural Language Toolkit), Keras and Tkinter library (for coding Graphical User Interface)

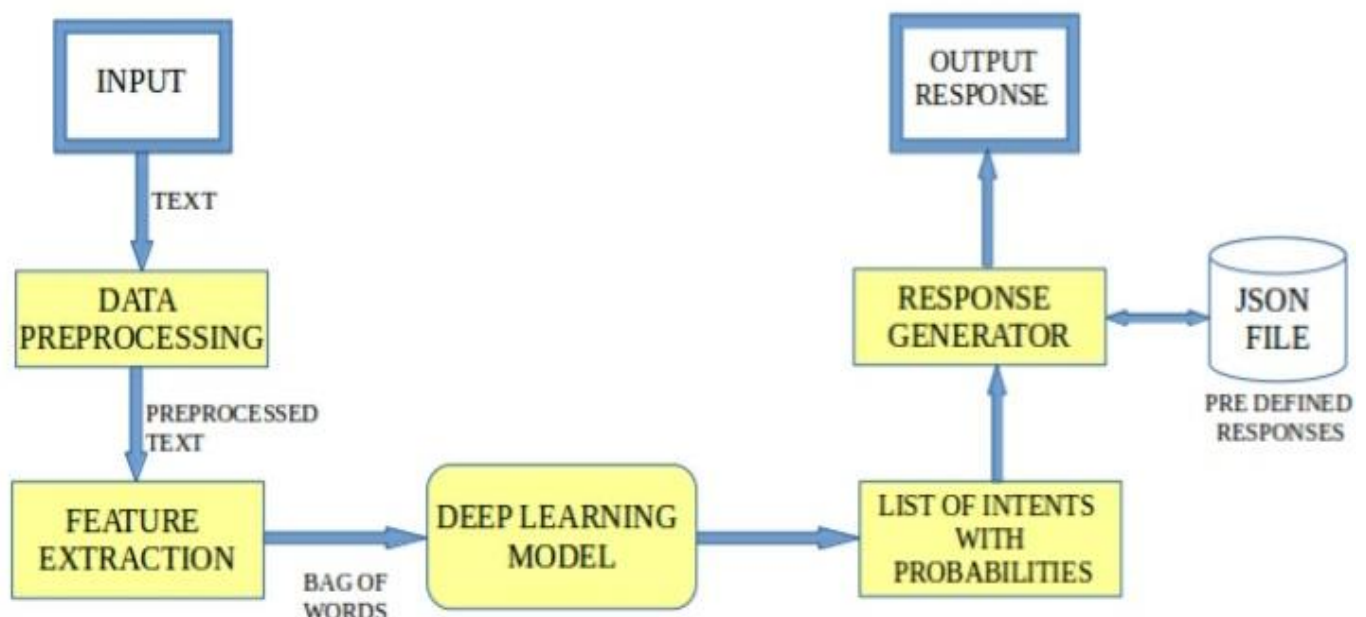


Fig. 3. System Overview

## Dataset:

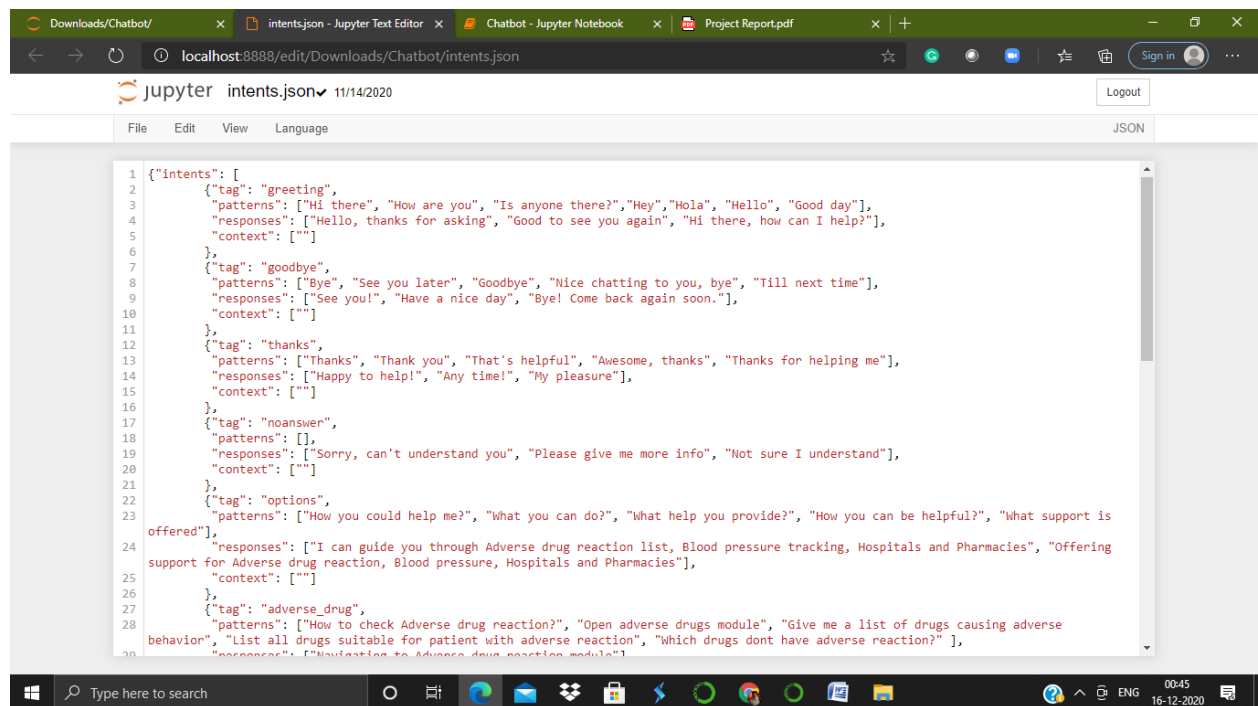
The dataset we will be using is 'intents.json'. This is a JSON file that contains the patterns we need to find and the responses we want to return to the user.

## Steps Involved:

### 1. Importing and loading the data file:

We will import all the necessary libraries and the data file. The data file is in JSON format so we will also use the json package to parse the JSON file into Python

This is how our intents file look like:



```
1 {"intents": [
2   {
3     "tag": "greeting",
4     "patterns": ["Hi there", "How are you", "Is anyone there?", "Hey", "Hola", "Hello", "Good day"],
5     "responses": ["Hello, thanks for asking", "Good to see you again", "Hi there, how can I help?"],
6     "context": [""]
7   },
8   {
9     "tag": "goodbye",
10    "patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye", "Till next time"],
11    "responses": ["See you!", "Have a nice day", "Bye! Come back again soon."],
12    "context": [""]
13  },
14  {
15    "tag": "thanks",
16    "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks", "Thanks for helping me"],
17    "responses": ["Happy to help!", "Any time!", "My pleasure"],
18    "context": [""]
19  },
20  {
21    "tag": "noanswer",
22    "patterns": [],
23    "responses": ["Sorry, can't understand you", "Please give me more info", "Not sure I understand"],
24    "context": [""]
25  },
26  {
27    "tag": "options",
28    "patterns": ["How you could help me?", "What you can do?", "What help you provide?", "How you can be helpful?", "What support is offered"],
29    "responses": ["I can guide you through Adverse drug reaction list, Blood pressure tracking, Hospitals and Pharmacies", "Offering support for Adverse drug reaction, Blood pressure, Hospitals and Pharmacies"],
30    "context": [""]
31  },
32  {
33    "tag": "adverse_drug",
34    "patterns": ["How to check Adverse drug reaction?", "Open adverse drugs module", "Give me a list of drugs causing adverse behavior", "List all drugs suitable for patient with adverse reaction", "Which drugs dont have adverse reaction? "],
35    "responses": ["Redirecting to Adverse drug reaction module"]
36  }
37 ]}
```

### 2. Preprocessing the data:

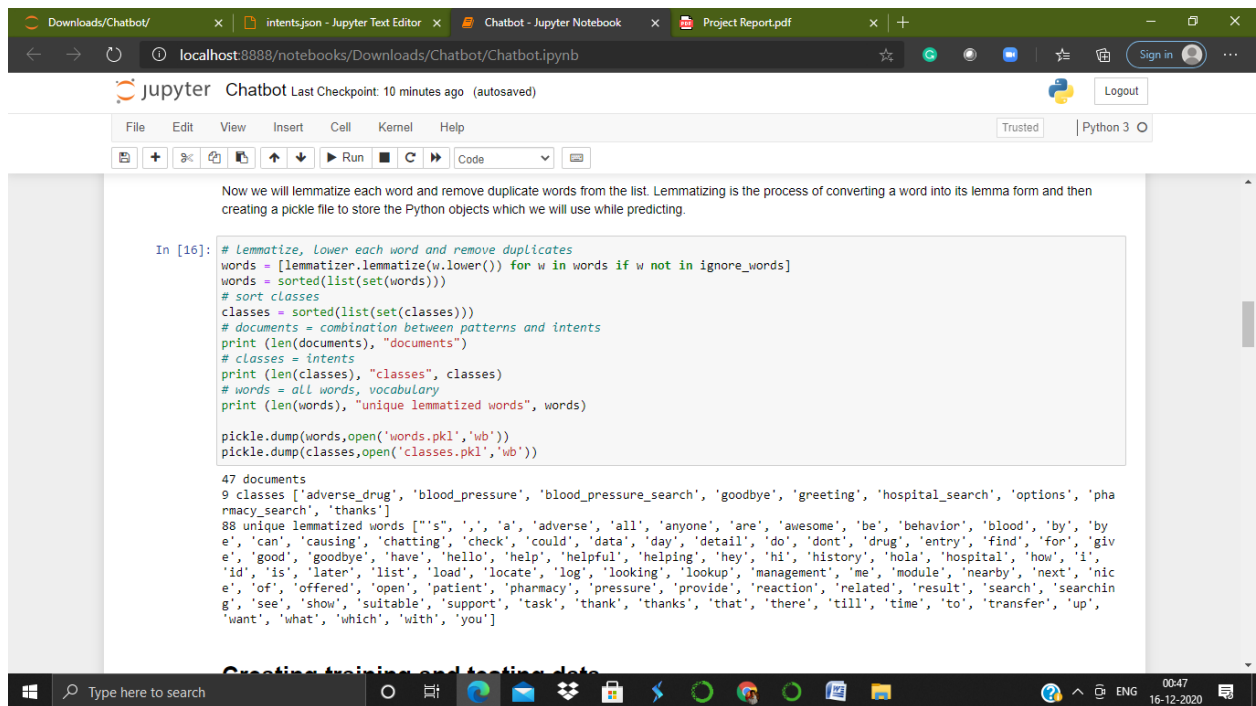
When working with text data, we need to perform various preprocessing on the data before we make a machine learning or a deep learning model. Tokenizing is the most basic and first thing we can do on text

data. Tokenizing is the process of breaking the whole text into small parts like words.

In this project we will iterate through the patterns and tokenize the sentence using `nltk.word_tokenize()` function and append each word in the words list. We will also create a list of classes for our tags and save the file as 'classes.pkl'.

After that we will lemmatize each word and remove duplicate words from the list. Lemmatizing is the process of converting a word into its lemma form and then creating a pickle file ( 'words.pkl' ) to store the Python objects which we will use while predicting.

List of classes and lemmatized words:



The screenshot shows a Jupyter Notebook window with the following content:

```
Now we will lemmatize each word and remove duplicate words from the list. Lemmatizing is the process of converting a word into its lemma form and then creating a pickle file to store the Python objects which we will use while predicting.
```

```
In [16]: # Lemmatize, Lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))
# sort classes
classes = sorted(list(set(classes)))
# documents = combination between patterns and intents
print(len(documents), "documents")
# classes = intents
print(len(classes), "classes", classes)
# words = all words, vocabulary
print(len(words), "unique lemmatized words", words)

pickle.dump(words, open('words.pkl', 'wb'))
pickle.dump(classes, open('classes.pkl', 'wb'))
```

47 documents  
9 classes ['adverse\_drug', 'blood\_pressure', 'blood\_pressure\_search', 'goodbye', 'greeting', 'hospital\_search', 'options', 'pharmacy\_search', 'thanks']  
88 unique lemmatized words ['a', 'adverse', 'all', 'anyone', 'are', 'awesome', 'be', 'behavior', 'blood', 'by', 'can', 'causing', 'chatting', 'check', 'could', 'data', 'day', 'detail', 'do', 'dont', 'drug', 'entry', 'find', 'for', 'give', 'good', 'goodbye', 'have', 'hello', 'help', 'helpful', 'helping', 'hey', 'hi', 'history', 'hola', 'hospital', 'how', 'i', 'id', 'is', 'later', 'list', 'load', 'locate', 'log', 'looking', 'lookup', 'management', 'me', 'module', 'nearby', 'next', 'nice', 'of', 'offered', 'open', 'patient', 'pharmacy', 'pressure', 'provide', 'reaction', 'related', 'result', 'search', 'searching', 'see', 'show', 'suitable', 'support', 'task', 'thank', 'thanks', 'that', 'there', 'till', 'time', 'to', 'transfer', 'up', 'want', 'what', 'which', 'with', 'you']

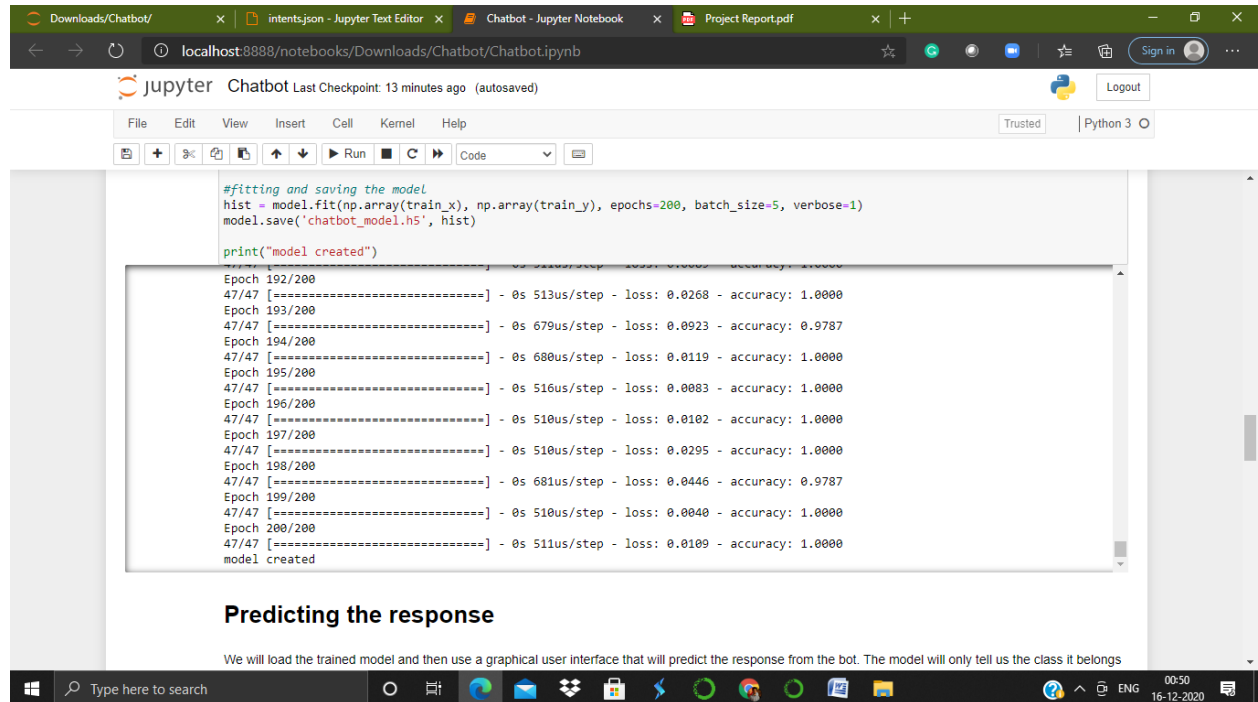
### 3. Creating training and testing data:

Then, we will create the training and testing data in which we will provide the input and the output. Our input will be the pattern and output will be the class our input pattern belongs to.

## 4. Building the Model:

After creating the training and testing data we will create a deep learning model with several layers. We will use the Keras sequential API for this. After training the model for several epochs to achieve maximum accuracy, we will save our model as 'chatbot\_model.h5'.

Creating the model:



```
#fitting and saving the model
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
model.save('chatbot_model.h5', hist)
print("model created")
```

Epoch 192/200  
47/47 [=====] - 0s 513us/step - loss: 0.0268 - accuracy: 1.0000  
Epoch 193/200  
47/47 [=====] - 0s 679us/step - loss: 0.0923 - accuracy: 0.9787  
Epoch 194/200  
47/47 [=====] - 0s 680us/step - loss: 0.0119 - accuracy: 1.0000  
Epoch 195/200  
47/47 [=====] - 0s 516us/step - loss: 0.0083 - accuracy: 1.0000  
Epoch 196/200  
47/47 [=====] - 0s 510us/step - loss: 0.0102 - accuracy: 1.0000  
Epoch 197/200  
47/47 [=====] - 0s 510us/step - loss: 0.0295 - accuracy: 1.0000  
Epoch 198/200  
47/47 [=====] - 0s 681us/step - loss: 0.0446 - accuracy: 0.9787  
Epoch 199/200  
47/47 [=====] - 0s 510us/step - loss: 0.0040 - accuracy: 1.0000  
Epoch 200/200  
47/47 [=====] - 0s 511us/step - loss: 0.0109 - accuracy: 1.0000  
model created

### Predicting the response

We will load the trained model and then use a graphical user interface that will predict the response from the bot. The model will only tell us the class it belongs to

## 5. Predicting the response using Graphical User Interface:

We will load the trained model and then use a graphical user interface that will predict the response from the bot. The model will only tell us the class it belongs to, so we will implement some functions which will identify the class and then retrieve us a random response from the list of responses.

Now we will load the 'words.pkl' and 'classes.pkl' pickle files which we have created when we trained our model.

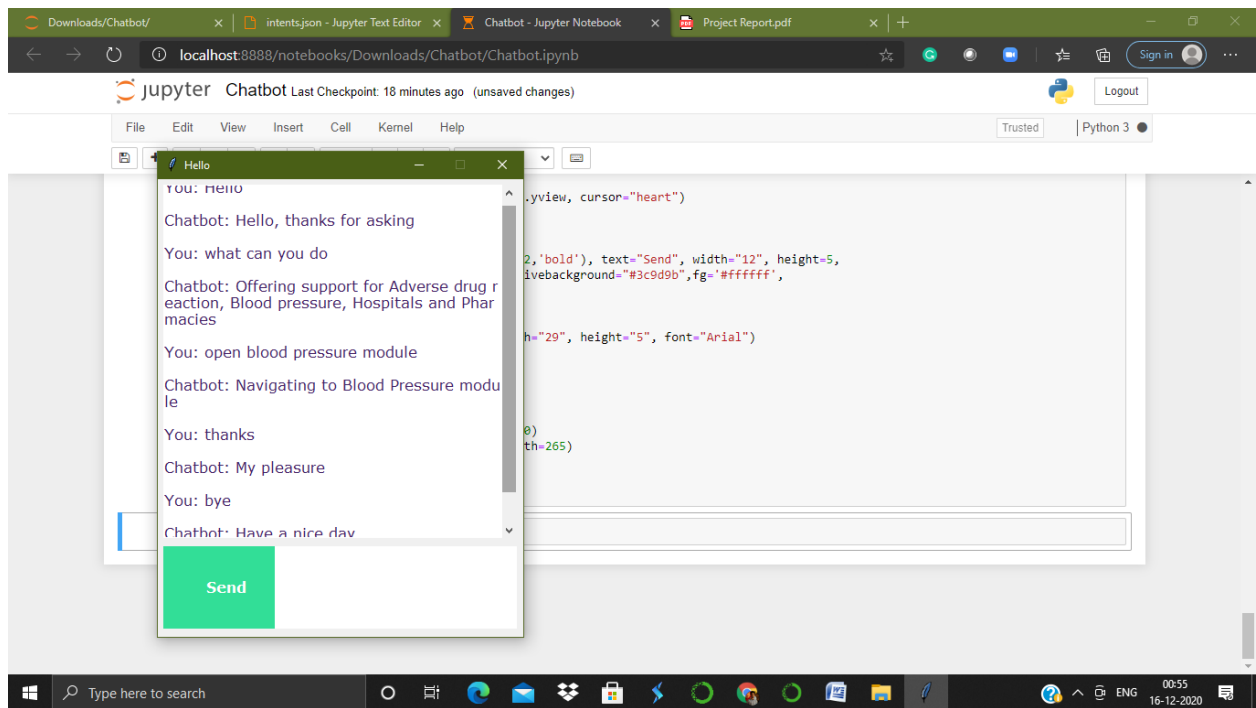
To predict the class, we will need to provide input in the same way as we did while training. So we will create some functions that will perform text preprocessing and then predict the class. After predicting the class, we will get a random response from the list of intents.

We will also code a graphical user interface. For this, we use the Tkinter library which already comes in python. We will take the input message from the user and then use the helper functions we have created to get the response from the bot and display it on the GUI.

## 6. Running the chatbot:

After running all the cells a Graphical User Interface window will open up within a few seconds. With the GUI we can easily chat with the bot

Conversation with Chatbot:



## Results:

In this project we have learnt about chatbots and implemented a Deep Learning version of a chatbot in Python.

## **Conclusion and Future scopes:**

Conversation AI is one of the growing research are in the field of Artificial Intelligence. The conversational engine can be used in any of the field for providing needed customer support. It is widely used in Educational Websites for querying about the Institutional Details, Medical field for taking doctor appointments, Business field for customer support etc. The proposed system is a retrieval based chatbot which works based on neural network. Deep Learning concept is used for the model building. As it is a retrieval based system, it particularly works in a closed domain.

Here we work on a Dataset of a Hospital Query and the system works well on that domain. Here when a user ask any query within the domain, the system will generate the corresponding response for it and drive the conversation in a meaningful way. Manual updation of the dataset is needed at regular intervals for improving the efficiency of the system and to help the system to answer most of the repeatedly asked queries of the user.

Apart from the current model, more improvements can be further added in order to increase the efficiency of the system. Some of the possible modifications or areas of improvements that can be done are mentioned here. Here for developing the model Keras NN is used, so there is a lot of room for improvement. Other type of neural networks such as Convolutional Network(CN) or Recurrent Network(RN) can also be used to make the system more efficient if needed and not only that Apart from keras framework, There are many more deep learning frameworks such as tensorflow, Apache Spark, PyTorch, Sonnet, and more which can be used for improving efficiency.

## **References:**

1. [NLTK Documentation](#)
2. [pickle library documentation](#)
3. [Tkinter library - documentation](#)
4. [Keras documentation](#)
5. [A publication on Retrieval Based Chatbots](#)
6. [A Quick Guide on Retrieval Based vs. Generative based chatbots](#)

