**Name**: Sandesh Shejwal

**Domain** : Data Science

**Batch** : B 4

**Project** : Chat bot

**ABSTRACT**

A chatbot is an artificial intelligence computer program which performs communication using audio and video system. A person can ask any questions and chatbot will answer accordingly. Nowadays a chatbot is highly popular and takes speed as a computer communication application. Chatbot system is in trend, thus it is being used on many websites. With the chatbot, one doesn’t have to wait to talk to the customer helpline, they don’t even have to search for shopping through Websites. A chatbot is used in many areas like order food, product suggestions, customer support, weather, personal finance assistance, scheduled a meeting, search and track flights, send money, and many more. The main objective that we will discuss in this paper is about creating a web API, and also about sample web and text messaging interfaces that demonstrate the use of API. In this research paper we are trying to understand these Chatbots and understand their shortcomings

**Applications**

A chatbot can be used anywhere a human is interacting with a computer system. These are the areas where the fastest adoption is occurring:

Customer Service: A chatbot can be used as an “assistant” to a live agent, increasing the agent’s efficiency. When trained, they can also provide service when the call centre is closed, or eventually even act as an independent agent, if desired.

**Sales/Marketing/Branding —**Chatbots can be used for sales qualification, ecommerce, promotional campaigns, or as a branding vehicle.

**Human Resources —**An HR chatbot can help with frequently asked questions (“how many vacation days do I have left?”) and can act as an onboarding assistant.

**Objective :**

Creating a chatbot using python and using various python libraries. Making use of modules like

* import random
* import json
* import pickle
* import numpy as np
* import nltk
* from keras.models import load\_model
* from nltk.stem import WordNetLemmatizer

for making the chat bot.

**INTRODUCTION :**

Chat has become the center of focus in this current era, thus the bots are being utilized to deliver information engagingly and conveniently. A chatbot is standout amongst the most progressive and promising tools of communication among people and machines. Famous chatbots like Google Assistant, Amazon Alexa, Siri, Facebook, Slack, and many more are in trend. These are very much helpful, but in this era of enhancing technology, day by day technology gets updated, and accordingly, user expectations also increase. A user wants more automation in the chatbot. Although every system is not perfect there is always a flaw in the system, so as in the chatbot there are some problems that the user has experienced while using a chatbot. Chatbot can be described as an answering system where a system will be able to answer questions or statements submitted by users and allow users to control over the content to be displayed. A bot is trained on and according to the training, based on some rules on which it is trained, it answers questions. It is called ruled based approach. Using these ruled based approach, creation of these bots becomes relatively straight forward. But it is not sufficient for the bot to answer questions whose pattern does not match with the rules on which it is trained. The language by which these bots can be created are Artificial Intelligence Markup Language(AIML). It is a language based on XML which allows the developer to write the rules which bot will follow.

**METHODOLOGY:**

We will follow a step-by-step approach and break down the procedure of creating a Python chat.

We will begin building a Python chatbot by importing all the required packages and modules necessary for the project. We will also initialize different variables that we want to use in it.

Now we will use our training and testing datasets , we will import it using the pickle library(convert it into readable format).

After that we will tokenize our data. tokenizing supports text data - it converts the large text dataset into smaller, readable chunks (such as words). Once this process is complete, we can go for lemmatization to transform a word into its lemma form. Then it generates a pickle file in order to store the objects of Python that are utilized to predict the responses of the bot.

* [**random**](https://www.geeksforgeeks.org/python-random-module/) – This module is used to generate random responses from the Chatbot
* [**json**](https://www.geeksforgeeks.org/python-json/) – To read from json file
* [**pickle**](http://www.geeksforgeeks.org/pickle-python-object-serialization/) – To save data into files
* **tensorflow** – To train neural networks. It is an open source machine learning library.
* [**numpy**](https://www.geeksforgeeks.org/numpy-in-python-set-1-introduction/) – It is a Python library used for working with arrays
* **nltk** – It is a leading platform for building Python programs to work with human language data.

Create a WordNetLemmatizer() class object.

Read the contents from the “intense.json” file and store it to a variable “intents”. Next, initialize empty lists to store the contents.

Next up, we have a function called word\_tokenize(para). It takes a sentence as a parameter and then returns a list containing all the words of the sentence as strings. Here we’re tokenizing the patterns and then appending them to a list ‘words’. So, at last, this list ‘words’ would have all the words that are in the ‘patterns’ list.

In documents, we have all the patterns with their tags in the form of a tuple.

Now, using a list comprehension, we’ll modify the list ‘words’ we created above and store the words’ ‘lemma’  or simply put, the root words.

Dump the data of the ‘words’ and ‘classes’ to binary files of the same name, using the pickle module’s dump() function.

Create an empty list called training, in which we’ll store the data used for training. Also create an output\_empty list that will store as many 0’s as there are classes in the intense.json.

Next up we’ll create a bag that will store the 0’s and 1’s. (0, if the word isn’t in the pattern and 1 if the word is in the pattern). To do that, we’ll iterate through the documents list and append 1 to the ‘bag’ if it is  not in the patterns, 0 otherwise.

Now shuffle this training set and make it a numpy array.

Split the training set consisting of 1’s and 0’s into two parts, that is train\_x and train\_y.

We’ve come to the model-building part of our Chatbot model. Here, we’re going to deploy a Sequential model, that we’ll train on the dataset we prepared above.

* Add():  This function is used to add layers in a neural network.
* Dropout(): This function is used to avoid overfitting
* **# required modules**
* **import** random
* **import** json
* **import** pickle
* **import** numpy as np
* **import** nltk
* **from** keras.models **import** load\_model
* **from** nltk.stem **import** WordNetLemmatizer

**CODE:**

import pickle

import numpy as np

with open('train\_qa.txt' , "rb") as fp:

train\_data = pickle.load(fp)

with open('test\_qa.txt' , "rb") as fp:

test\_data = pickle.load(fp)

len(train\_data)

with open('test\_qa.txt' , "rb") as fp:

test\_data = pickle.load(fp)

len(test\_data)

train\_data[0][2]

vocab = set()

all\_data = test\_data + train\_data

for story , question , answer in all\_data:

vocab = vocab.union(set(story))

vocab = vocab.union(set(question))

vocab.add('yes')

vocab.add('no')

len(vocab)

vocab\_len = len(vocab) + 1

max\_story\_len = max([len(data[0]) for data in all\_data])

max\_story\_len

max\_ques\_len = max([len(data[1]) for data in all\_data])

max\_ques\_len

from keras\_preprocessing.sequence import pad\_sequences

from keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer(filters = [])

tokenizer.fit\_on\_texts(vocab)

tokenizer.word\_index

# train dataset

train\_story\_text = []

train\_question\_text = []

train\_answers = []

for story , question , answer in train\_data:

train\_story\_text.append(story)

train\_question\_text.append(question)

train\_story\_seq = tokenizer.texts\_to\_sequences(train\_story\_text)

train\_story\_seq

def vectorize\_stories(data , word\_index = tokenizer.word\_index, max\_story\_len = max\_story\_len , max\_ques\_len = max\_ques\_len ):

X = []

Xq =[]

Y =[]

for story , query , answer in data :

x = [word\_index[word.lower()] for word in story]

xq = [word\_index[word.lower()] for word in query]

y = np.zeros(len(word\_index) + 1)

y[word\_index[answer]] = 1

X.append(x)

Xq.append(xq)

Y.append(y)

return(pad\_sequences(X,maxlen=max\_story\_len),

pad\_sequences(Xq,maxlen=max\_ques\_len),

np.array(Y) )

inputs\_train , queries\_train, answers\_train = vectorize\_stories(train\_data)

inputs\_test , queries\_test , answers\_test = vectorize\_stories(test\_data)

inputs\_train

queries\_test

answers\_test

tokenizer.word\_index['yes']

tokenizer.word\_index['no']

from keras.models import Sequential, Model

from keras.layers import Embedding

from keras.layers import Input , Activation , Dense , Permute , Dropout , add, dot , concatenate , LSTM

input\_sequence = Input((max\_story\_len,))

question = Input((max\_ques\_len,))

#input encoder m

input\_encoder\_m = Sequential()

input\_encoder\_m.add(Embedding(input\_dim = vocab\_len , output\_dim = 64))

input\_encoder\_m.add(Dropout(0.3))

#input encoder C

input\_encoder\_c = Sequential()

input\_encoder\_c.add(Embedding(input\_dim = vocab\_len , output\_dim = max\_ques\_len))

input\_encoder\_c.add(Dropout(0.3))

question\_encoder = Sequential()

question\_encoder.add(Embedding(input\_dim = vocab\_len , output\_dim = 64 , input\_length = max\_ques\_len))

question\_encoder.add(Dropout(0.3))

#Encode the sequences

input\_encoded\_m = input\_encoder\_m(input\_sequence)

input\_encoded\_c = input\_encoder\_c(input\_sequence)

question\_encoded = question\_encoder(question)

match = dot([input\_encoded\_m , question\_encoded ] , axes = (2,2))

match = Activation('softmax')(match)

response = add([match,input\_encoded\_c])

response = Permute((2,1))(response)

answer = concatenate([response,question\_encoded])

answer = LSTM(32)(answer)

answer = Dropout(0.5)(answer)

answer = Dense(vocab\_len)(answer)

answer = Activation('softmax')(answer)

model = Model([input\_sequence , question] , answer)

model.compile(optimizer = 'rmsprop' , loss = 'categorical\_crossentropy' , metrics = ['accuracy'])

model.summary()

history = model.fit([inputs\_train , queries\_train] , answers\_train,

batch\_size = 32, epochs = 20,

validation\_data = ([inputs\_test , queries\_test],answers\_test)

)

import matplotlib.pyplot as plt

print(history.history.keys())

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title("Model Accuracy")

#save

model.save("chatbot\_modelDS")

#evaluate the test set

model.load\_weights("chatbot\_modelDS")

pred\_results = model.predict(([inputs\_test , queries\_test]))

test\_data[0][0]

story = ' '.join(word for word in test\_data[13][0])

story

query = ' '.join(word for word in test\_data[13][1])

query

test\_data[13][2]

val\_max = np.argmax(pred\_results[13])

for key , val in tokenizer.word\_index.items():

if val == val\_max:

k = key

print("Predicted Answer is" , k)

print("Probability of Certanity" , pred\_results[13][val\_max])

**CONCLUSION:**

A database is used in many applications for the connection of Chatbots. Every customer or user needs appropriate answers and so database is used to so that purpose can be solved. The primary point here is that smart bots can help increase the customer base by enhancing the customer support services, thereby helping to increase sales.

Successfully created chat bot using python. Made use of train and test datasets. Made use of various python libraries and modules.