# Python Chatterbot

T&T Lab Project

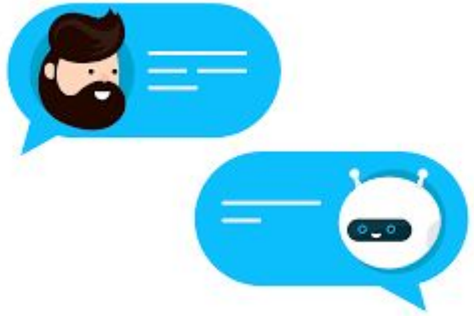
Prepared By-
Madhurima Choudhury (1805745)
Shubhangi Srivastava (1805706)

# What is a ChatBot?

A **chat bot** is software that conducts conversations. Many chatbots are created to simulate how a human would behave as a conversational partner. Chat bots are in many devices, for example Siri, Cortona, Alexa, and Google Assistant. Many chat bots are used now a days for customer service.

There are broadly two variants of chat bots: **Rule-Based** and **Self Learning**. A Rule-Based chatbot is a bot that answers questions based on some rules that it is trained on, while a Self Learning chat bot is a chat bot that uses some Machine Learning based technique to chat.

We will use a rule based approach for responding back to greetings, and we will have the chat bot respond to questions and queries by taking in some text and having the chat bot select the best response back from that text. This type of self learning is called **retrieval-based** learning.



## Prediction in the need of Chickenpox Vaccination

**Source**: Centre of Disease Control and Detection , US Dept of Health & Human Services.

**Link for Web Scraping:**
https://www.cdc.gov/vaccines/vpd/varicella/hcp/administering-vaccine.html

Our Doctor Bot will greet and respond to the user's query regarding What everyone should know about Chicken pox vaccination, including it's requirements, who needs the vaccine and who doesn't, types of the vaccine and so on, using Natural Language Processing.

# Installation of the required packages

nltk

newspaper3k

**Natural Language Toolkit**

Natural Language Processing with Python NLTK is one of the leading platforms for working with human language data and Python, the module NLTK is used for natural language processing.

The Newspaper3k package is a Python library used for Web Scraping articles,.

This module is a modified and better version of the *Newspaper* module which is also used for the same purpose.

## Importing required Libraries

### Article from Newspaper3k

```
from newspaper import
Article
```

We will use the newspaper library to extract the text from the website by using the Article class.

### Random

Python defines a set of function that are used to generate or manipulate random strings or numbers through Random Module.

In our Python Chatterbot, we've used the inbuilt function , **choice( )**, that returns random item from the provided list as a Bot's greeting response

### Other Libraries-

- String: Since we're involved in the manipulation of the ASCII_letters (lowercase +uppercase); so string library is imported
- Numpy: general purpose array-processing package for working with the multi - dimensional array.
- NLTK: The Natural Language Toolkit (NLTK) is a Python package for natural language processing

# Importing required Libraries

---continued

## CountVectorizer

```
from
sklearn.feature_extraction.text
import CountVectorizer
```

CountVectorizer, a great tool provided by the skit library ; used to transform a given text into a vector/matrix in which each unique word from a given text forms the column and each text sample from the is the row in the matrix. The value of each cell is the count of the word in that particular text sample.

## Cosine_similarity

```
from sklearn.metrics.pairwise
import cosine_similarity
```

From the sklearn.metrics.pairwise library we will get the cosine_similarity method to see how similar the text is to the users queries.

## Warnings

In Python, there are a variety of built-in exceptions, and one of its subclasses are the Warning Modules.

```
warnings.filterwarnings('
ignore')
```

Here, we've performed the action-'ignore' on any parameter to filter out the unnecessary warnings produced in the code-execution

## STEP1: Article Extraction & Tokenization

```
#Downloading the Punkt Package
nltk.download('punkt',quiet=True)

#Extracting the desired Article
article=Article('https://www.cdc.gov/vaccin
es/vpd/varicella/public/index.html')
article.download()
article.parse()
article.nlp()
corpus=article.text

# Tokenization
text=corpus
sentence_list=nltk.sent_tokenize(text)
print(sentence_list)
```

- **The Punkt-nltk Sentence Tokenizer:** divides a text into a list of sentences, by using an unsupervised algorithm to build a model for abbreviation words, collocations, and words that start sentences.
- **Article Extraction:** Here we've extracted the data from the provided url of a government website related to Chickenpox Symptoms and Vaccination, and made it suitable for tokenization.
- **Tokenization :** Here we're tokenizing/splitting the whole data into a list of tokens(sentences) , by using the sent_tokenize( ) function from the NLTK's punkt module , which is already been trained and thus very well knows to mark the end and the beginning of the sentences and what characters and punctuations.

# STEP 2: Returning a random greeting response to the user's greetings

```
def greeting_response(text):
  text=text.lower()
      # Bot's Greeting response
   bot_greetings=['hello','hi','hey','hola','howdy']
      # User's Greeting
   user_greetings=['hello','hi','hey','hola','wassup']

  for word in text.split():
    if word in user_greetings:
      return random.choice(bot_greetings)
```

Next we will use keyword matching (a rule based approach) to check for greeting type words as input from the user and respond back with a randomized greeting as output.

To do this we need to create a list of greeting inputs (the greetings we expect from the user) and then we will create a list of greeting responses (the greetings that our chat bot will use).

Then we will create a function to check for the users greetings and randomly choose a greeting response back.

## STEP 3: Returning the indices of the sorted values from the array

```python
def index_sort(list_var):
  length=len(list_var)
  list_index=list(range(0,length))

  x=list_var
  for i in range(length):
    for j in range(length):
      if x[list_index[i]]>x[list_index[j]]:
        #Swap
        temp=list_index[i]
        list_index[i]=list_index[j]
        list_index[j]=temp

  return list_index
```

A function index_sort() is created  to return the indices of the values from an array in sorted order by the arrays values. This function will help return the chatbot response.

## STEP4: Create Bot's Response

```
def bot_response(user_input):
      user_input=user_input.lower()
      sentence_list.append(user_input)
      bot_response=''
      cm=CountVectorizer().fit_transform(sentence_list)
      similarity_scores=cosine_similarity(cm[-1],cm)

      similarity_scores_list=similarity_scores.flatten()
      index=index_sort(similarity_scores_list)
      index=index[1:]
      response_flag=0
      j=0
      for i in range(len(index)):
        if similarity_scores_list[index[i]]>0.0:
         bot_response=bot_response+' '+sentence_list[index[i]]
         response_flag=1
         j=j+1
       if j>2:
         break

     if response_flag==0:
        bot_response=bot_response+" "+"I apologize I don't
      understand"

      sentence_list.remove(user_input)
      return bot_response
```

We are going to create a function bot_response() which will take in a users response or queries, and then send back the best response(s) selected from the corpus. The steps to be followed are-

- ➔ Convert the user's input to all lowercase letters
- ➔ Append the users response to the list of sentence tokens
- ➔ Create an empty response for the bot
- ➔ Create the count matrix
- ➔ Get the similarity scores to the users input
- ➔ Reduce the dimensionality of the similarity scores
- ➔ Get all of the similarity scores except the first (the query itself)
- ➔ Set a flag letting us know if the text contains a similarity score greater than 0.0
- ➔ Loop the through the index list and get the 'n' number of sentences as the response
- ➔ if no sentence contains a similarity score greater than 0.0 then print 'I apologize, I don't understand'
- ➔ Remove the users response from the sentence tokens

## STEP 5: Start the Chat

```
print("Doc Bot: Hello, I'm Doctor Bot or Doc Bot
for short.I'll answer your queries about Chickenpox
Vaccination. If you want to exit type-Bye!")

exit_list=['exit','see you
later','break','bye','quit']

while(True):
  user_input=input()
  if user_input.lower()in exit_list:
    print('Doc Bot: Chat with you later!')
    break

  else:
    if greeting_response(user_input)!= None:
      print("Doc Bot:
"+greeting_response(user_input))
    else:
      print('Doc Bot: '+bot_response(user_input))
```

Firstly, our Doctor Bot or Doc Bot will introduce itself to the user and will ask whether the user wants to continue or not.We can now create a continuous loop for the chatbot to converse with the user. We will run this loop until the users response is 'exit'.

CASE I: TERMINATION - Next we will use keyword matching (a rule based approach) to check for exit type words as input from the user and respond back with a "Chat with you later" as output. To do this we need to create an exit_list inputs (the termination words, which we expect from the user) Then we will create a function to check for the users greetings and randomly choose a greeting response back.

CASE II: GREETINGS- If the user greets the bot, then we'll call the greeting_response() & our Doc Bot will respond immediately with a randomized greetings, provided in the function.

CASE III: Information Gathering: If the user asks about any other information, then it'll call the bot_response() function and according to that it'll provide the necessary details to the user.

Thank You!