CREATE A CHATBOT IN PYTHON

AI-PHASE 3

Name: k. sneka

Reg.No: 513521106035

Department: ECE

Year: III

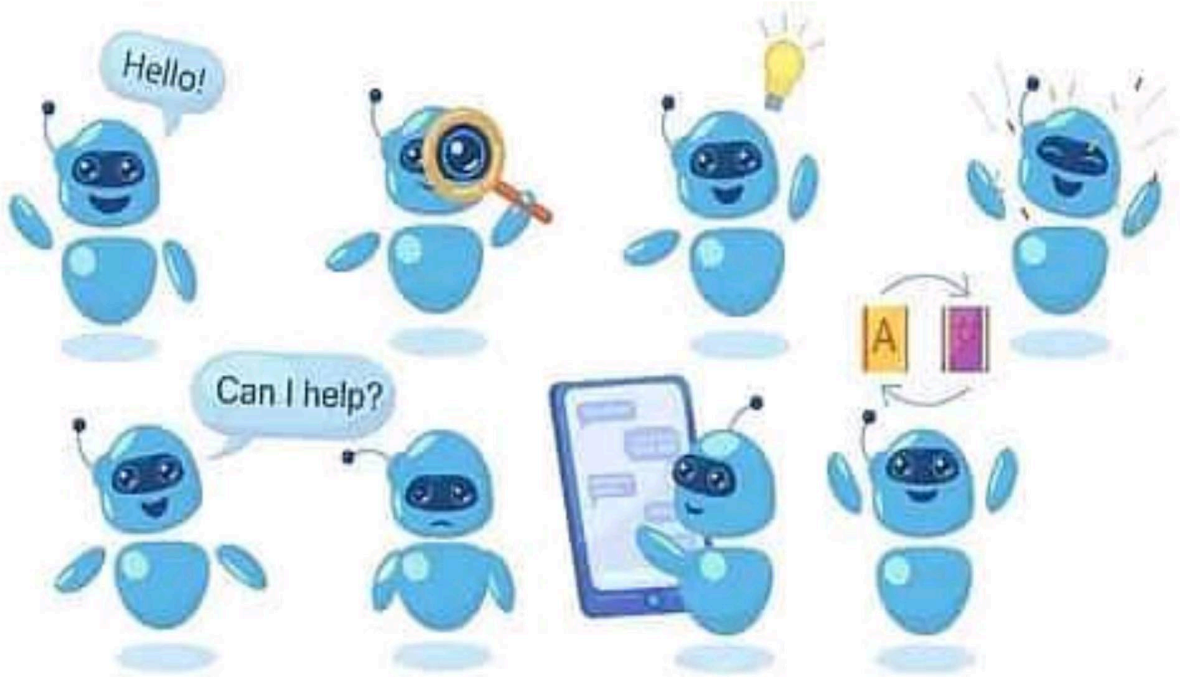
NM ID au513521106035

Email: smileysneha475@gmail.com

INTRODUCTION:

Natural Language Processing (NLP)- the exciting subdomain of Artificial Intelligence that deals with the interaction between computers and humans using the natural language. In this python chatbot

use exciting NLP libraries and learn how to make a chatbot from scratch in Python.

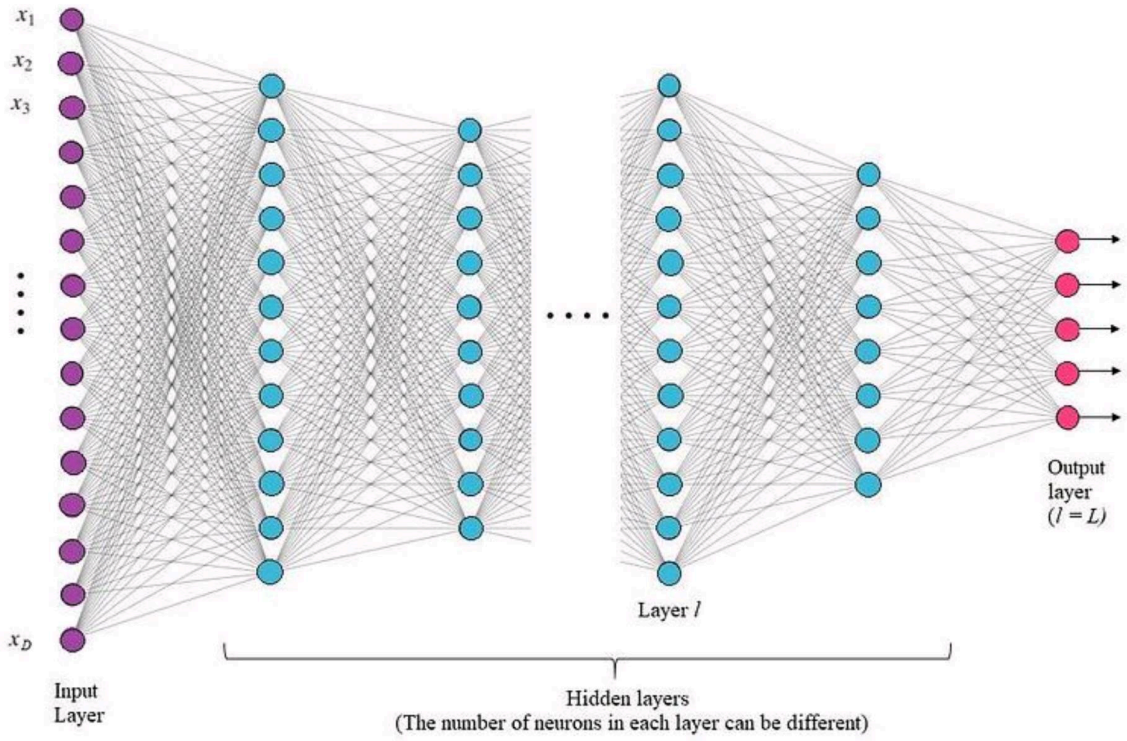


NEURAL NETWORK:

It is a deep learning algorithm that resembles the way neurons in our brain process information (hence the name). It is widely used to realize the pattern between the input features and the corresponding output in a dataset. Here is the basic neural network architecture.

The purple-colored circles represent the input vector, xi where i =1, 2, ….., D which is nothing else but one feature of our dataset. The blue-colored circles are the neurons of hidden layers. These are the layers that will learn the math required to relate our input with the output. Finally, we have pink-colored circles which form the output layer. The dimension of the output layer depends on the number of different classes that we have. For example, let us say we have 5x4 sized dataset where we have 5 input vectors, each having some

value for 4 features: A, B, C, and D. Assume that we want to classify each row as good or bad and we use the number 0 to represent good and 1 to represent bad. Then, the neural network is supposed to have 4 neurons at the input layer and 2 neurons at the output.



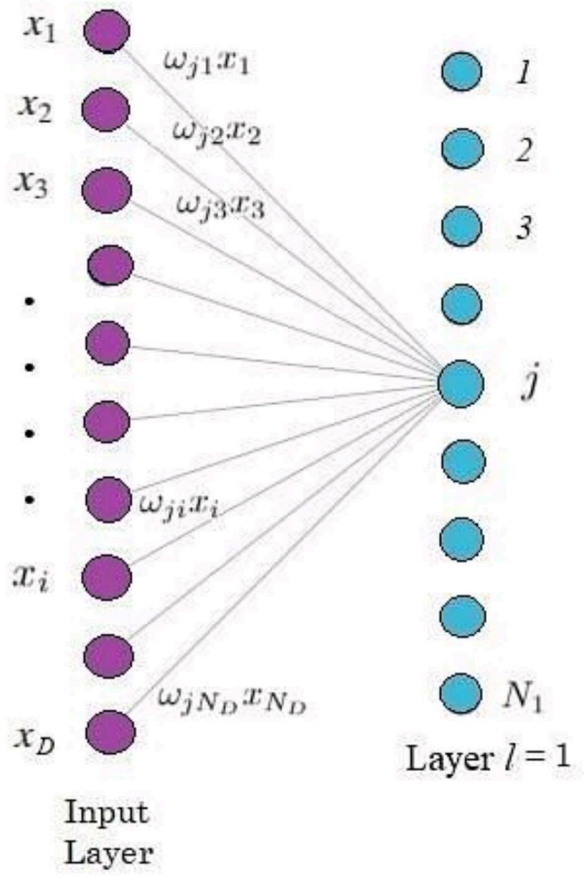
Neural Network algorithm involves two steps:

1. Forward Pass through a Feed-Forward Neural Network
2. Backpropagation of Error to train Neural Network

1.Forward Pass through a Feed-Forward Neural Network:

This step involves connecting the input layer to the output layer through a number of hidden layers. The neurons of the first layer (l=1) receive a

weighted sum of elements of the input vector (xáµ¢) along with a bias term b, as shown in Fig. 2. After that, each neuron transforms this weighted sum received at the input, a, using a differentiable, nonlinear activation function h(·) to give output z.



$$z_j^{(1)} = h(a_j^{(1)})$$
$$= h(\sum_{i=1}^{D} \omega_{ji}^{(1)} x_i + b_j^{(1)})$$

For a neuron of subsequent layers, a weighted sum of outputs of all the neurons of the previous layer along with a bias term is passed as input. This has been represented in Fig. 3. The layers of the subsequent layers to transform the input received using activation functions.

This process continues till the output of the last layer's (l=L) neurons has been evaluated. These neurons at the output layer are responsible for identifying the class the input vector belongs to. The input vector is labeled with the class whose corresponding neuron has the highest output value.

2.**Backpropagation of** Error to Train Neural Network:

This step is the most important one because the original task of the Neural Network algorithm is to find the correct set of weights for all the layers that lead to the correct output and this step is all about finding those correct weights and biases. Consider an input vector that has been passed to the network and say, we know that it belongs to class A. Assume the output layer gives the highest value for class B. There is therefore an error in our prediction. Now, since we can only compute errors at the output, we have to propagate this error backward to learn the correct set of weights and biases. Let us define the error function for the network:

$$E = \sum_{n=1}^{N} E_n$$

Here Eâ,™ is the error for a single pattern vector: xâ,™ and is defined as,

$$E_n = \frac{1}{2}\sum_{k}(z_k^{(L)} - r_k)^2$$

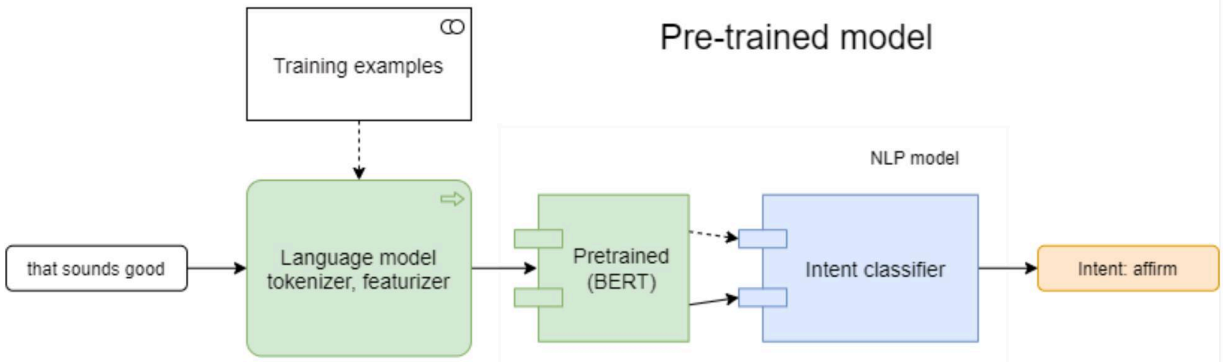
k = 1,2,3,..., NÊŸ; zâ,– is the output value of the káµ—Ê° neuron of the output layer L, and râ,– is the desired response for the káµ—*Ê° neuron of the output layer.*

NEURAL LANGUAGE PREPROCESSING:

At the beginning of the chatbot development process, however, you may face the lack of training data, which results in low accuracy in intent classification. A few workarounds exist to solve this problem:

1. Pre-trained model
2. Training data generator
3. Crowdsource

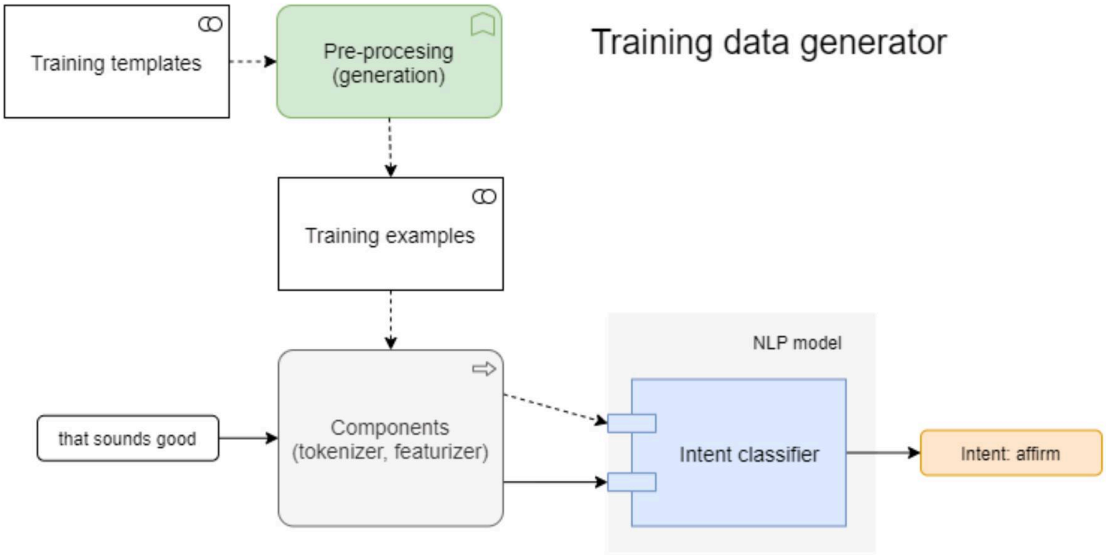
1.Pre-Trained model:



The pre-trained language model can be used for NLU tasks without any task-specific change to the model architecture. Pre-trained models have an ability to continue pre-training on custom data, strarting from some checkpoint.

This process is compute-intensive and requires a massively parallel compute infrastructure. The training of such general models on task-specific training data, called Fine-tuning, is far less computationally demanding and used more often. Well-known examples of similar models are Google's BERT, XLNet and OpenAI's GPT-2.

2. Training data generator



3. Crowdsource in AI chatbot development

To get more training data of high quality, you can outsource jobs to distributed workers, using Amazon Mechanical Turk, Microworkers, Clickworker platforms.

MTurk is one of the text preprocessing tools to take simple and repetitive tasks that need to be handled manually. In a short time and at a reduced cost, you can get human-written training data for your initial model to augment training data collection and ultimately accelerate Python-based chatbot development using data preprocessing for text classification.

1. *Import Necessary Libraries:*

   python

   import re

```
import string
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

2. *Lowercasing:*

   Convert all text to lowercase to ensure consistency in text analysis.

```
python
text = text.lower()
```

3. *Tokenization:*

   Tokenize the text into words or sentences.

```
python
words = word_tokenize(text)
```

4. *Remove Punctuation:*

   Remove punctuation marks from the text.

```
python

words = [word for word in words if word.isalnum()]
```

5. *Remove Stop Words:*

   Remove common stop words (e.g., "the," "and," "is") that don't contribute much to the meaning.

```
python

stop_words = set(stopwords.words('english'))

words = [word for word in words if word not in stop_words]
```

6. *Stemming or Lemmatization (Optional):*

   Reduce words to their base or root form to improve text normalization.

```
python

# Using NLTK for stemming

from nltk.stem import PorterStemmer

stemmer = PorterStemmer()
```

```
words = [stemmer.stem(word) for word in words]


# Using NLTK for lemmatization
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
words = [lemmatizer.lemmatize(word) for word in words]
```

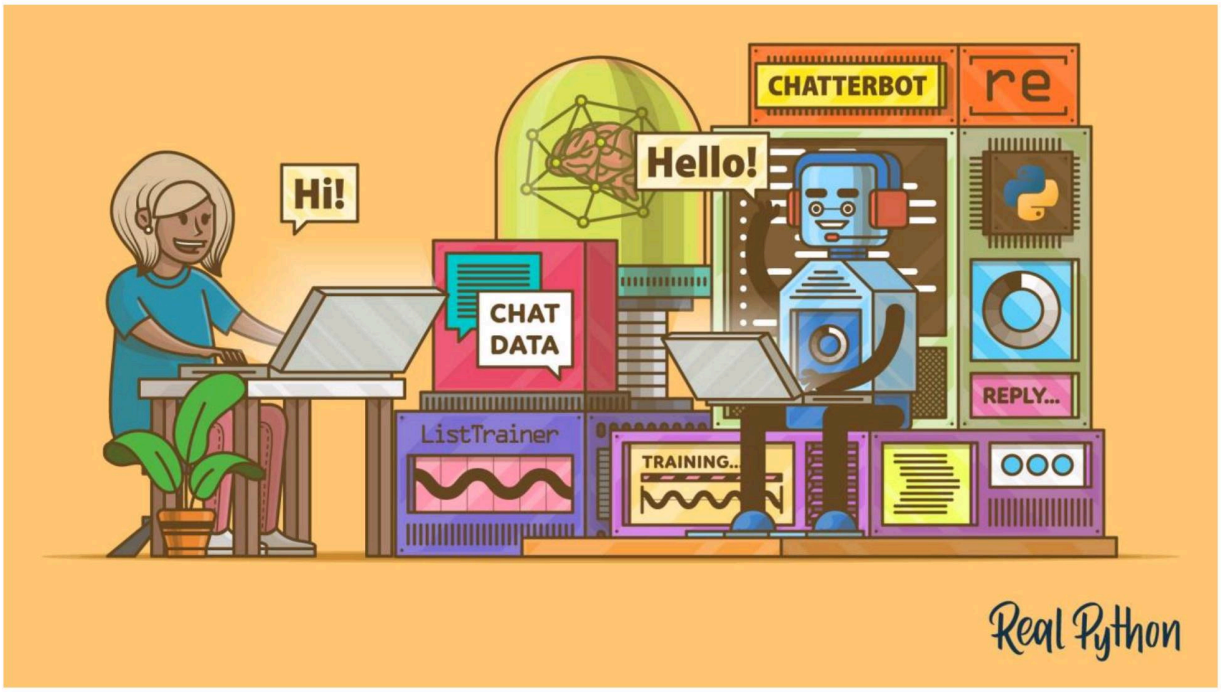
7. *Custom Cleaning (if needed):*

   Depending on your specific use case, you might want to perform additional cleaning steps like handling special characters or domain-specific terms.

8. *Rejoin Text:*

   Rejoin the words into a cleaned text string.

```
python
cleaned_text = ' '.join(words)
```



Chatbots can provide real-time customer support and are therefore a valuable asset in many industries. When you understand the basics of the **ChatterBot** library, you can **build and train** a **self-learning chatbot** with just a few lines of Python code.

You'll get the basic chatbot up and running right away in step one, but the most interesting part is the learning phase, when you get to train your chatbot. The quality and preparation of your training data will make a big difference in your chatbot's performance.

To simulate a real-world process that you might go through to create an industry-relevant chatbot, you'll learn how to customize the chatbot's responses. You'll do this by preparing **WhatsApp chat data** to train the chatbot. You can apply a similar process to train your bot from different conversational data in any domain-specific topic.

THANK YOU