**CHATBOT USING PYTHON**

TEAM MEMBER

963521106006: AKASH.K

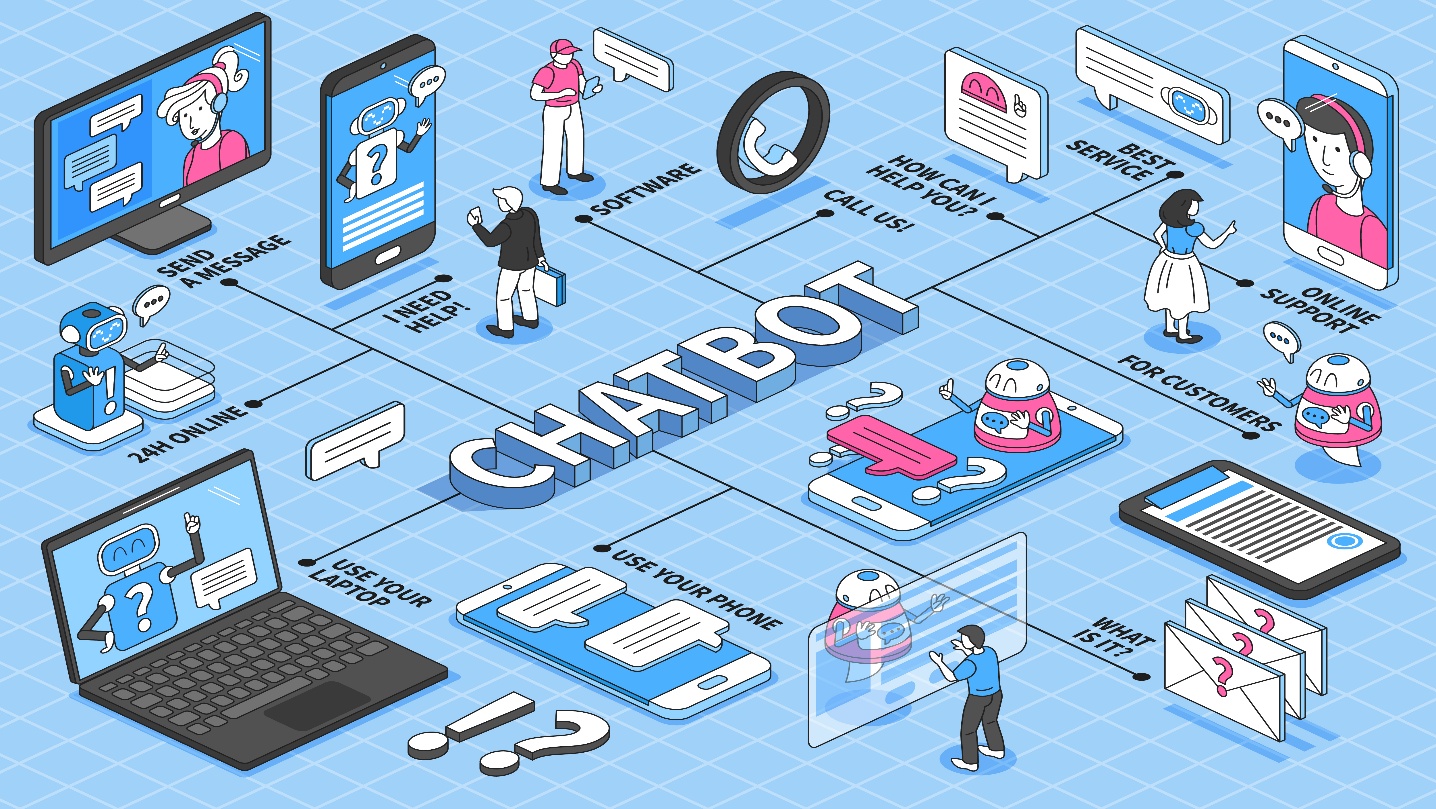
PHASE 3

PROJECT DEVELOPMENT PART 1

**TITLE: Create a Chatbot using Python**

**Abstract:**

Chatbots are computer programs that can simulate conversation with humans. They are becoming increasingly popular in a variety of applications, such as customer service, education, and entertainment. Python is a popular programming language for chatbot development, due to its flexibility and ease of use. Chatbots are computer programs that can simulate conversation with humans.

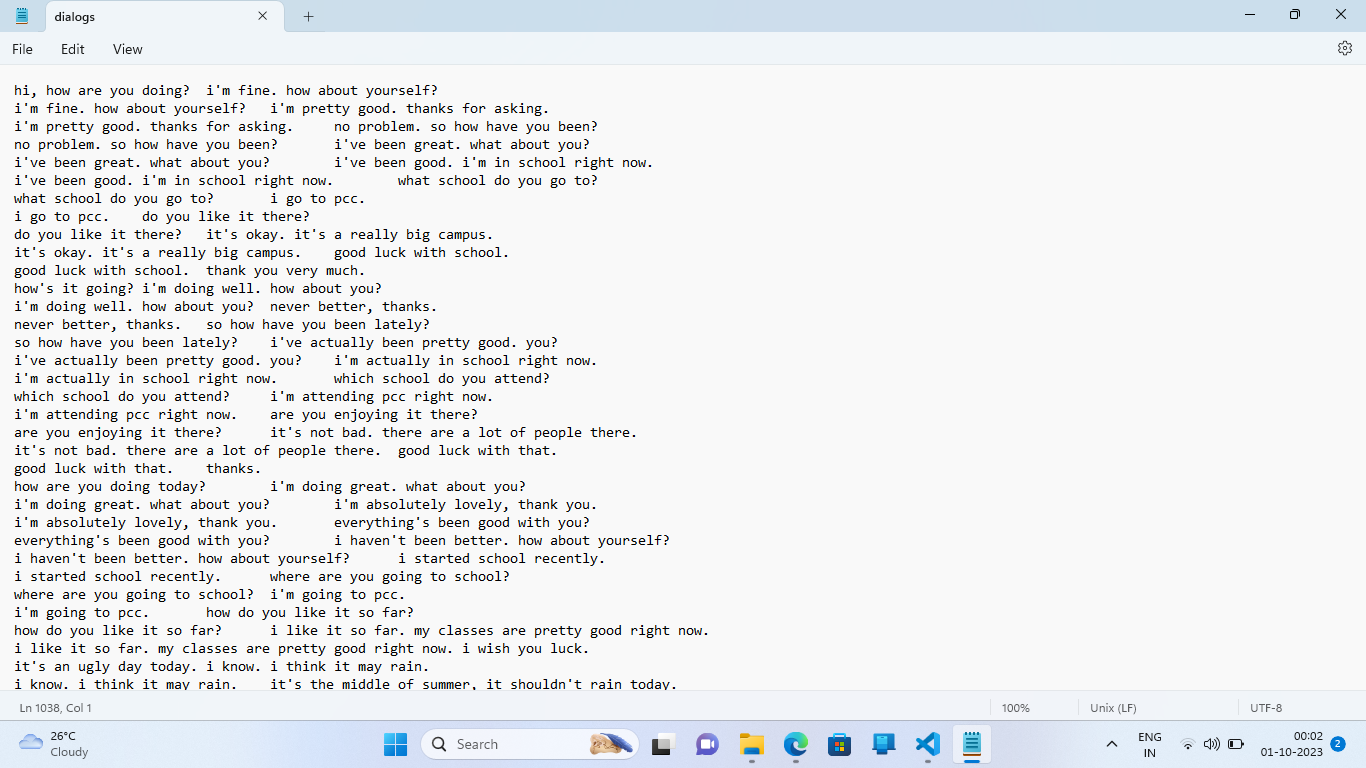


**Problem Definition:**

The challenge is to create a Chatbot in Python that provides exceptional customer service, answering user queries on a website or application. The objective is to deliver high-quality support to users, ensuring a positive user experience and customer satisfaction.

**Dataset:**

**Dataset Link:** [**https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot**](https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot)



**LOADING THE DATASET:**

To load the dataset for creating a chatbot using Python, you can use the following steps:

1. **Identify the format of your dataset.** Your dataset can be in a variety of formats, such as CSV, JSON, or text file.
2. **Install the required libraries.** Depending on the format of your dataset, you may need to install additional Python libraries. For example, if your dataset is in CSV format, you will need to install the pandas library.
3. **Load the dataset into a Python object.** You can use the appropriate Python library function to load your dataset into a Python object. For example, to load a CSV file, you can use the pandas.read\_csv() function.

In [1]:

*#model*  
 *import* tensorflow as tf  
 from sklearn.model\_selection import train\_test\_split  
  
 #nlp processing  
 import unicodedata  
 import re  
 import numpy as np  
  
  
 import warnings   
 warnings.filterwarnings('ignore')

/opt/conda/lib/python3.10/site-packages/scipy/\_\_init\_\_.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5  
 warnings.warn(f"A NumPy version >={np\_minversion} and <{np\_maxversion}"

In [2]:

*#reading data*  
 *data*=open('/kaggle/input/simple-dialogs-for chatbot/dialogs.txt','r').read()

**PREPROCESSING THE DATASET:**

Preprocessing the dataset is an important step in chatbot training. It involves cleaning and converting the data into a format that can be used by the chatbot model.

## **Segmentation**

Segmentation is the process of dividing a dataset into different groups based on shared characteristics. This can be a useful technique for chatbot development, as it allows you to train different chatbot models for different user groups.

For example, you could segment your dataset by user demographics, such as age, gender, and location. You could also segment your dataset by user interests, such as hobbies and products. Once you have segmented your dataset, you can train different chatbot models for each group. This will allow you to provide more personalized and relevant responses to your users.

There are a number of different ways to segment a dataset for chatbot development. One common approach is to use natural language processing (NLP) techniques to extract features from the text data, such as keywords and phrases. You can then use these features to cluster the data into different groups.

Another approach is to use machine learning techniques to train a model to predict the user group for each data point. This model can then be used to segment the dataset into different groups.

Once you have segmented your dataset, you can train different chatbot models for each group. You can use any of the various Python chatbot libraries available, such as ChatterBot, Rasa, or TensorFlow.

Here are some of the benefits of using segmentation in creating chatbot using Python:

* **More personalized and relevant responses:** By training different chatbot models for different user groups, you can provide more personalized and relevant responses to your users.
* **Improved user satisfaction:** Users are more likely to be satisfied with a chatbot that provides personalized and relevant responses.
* **Increased user engagement:** Users are more likely to engage with a chatbot that provides them with the information and services they need.
* **Improved chatbot performance:** By training chatbot models on specific data sets, you can improve their performance.

In [3]:

*#paried list of question and corresponding answer*  
 *QA\_list*=[QA.split('**\t**') for QA **in** data.split('**\n**')]  
 print(QA\_list[:5])

[['hi, how are you doing?', "i'm fine. how about yourself?"], ["i'm fine. how about yourself?", "i'm pretty good. thanks for asking."], ["i'm pretty good. thanks for asking.", 'no problem. so how have you been?'], ['no problem. so how have you been?', "i've been great. what about you?"], ["i've been great. what about you?", "i've been good. i'm in school right now."]]

In [4]:

questions=[row[0] for row **in** QA\_list]  
 answers=[row[1] for row **in** QA\_list]

In [5]:

print(questions[0:5])  
 print(answers[0:5])

['hi, how are you doing?', "i'm fine. how about yourself?", "i'm pretty good. thanks for asking.", 'no problem. so how have you been?', "i've been great. what about you?"]  
["i'm fine. how about yourself?", "i'm pretty good. thanks for asking.", 'no problem. so how have you been?', "i've been great. what about you?", "i've been good. i'm in school right now."]

## **Normalization**

Normalization is the process of converting text into a consistent format. This is an important step in chatbot development, as it allows the chatbot model to better understand the text data.

There are a number of different normalization techniques that can be used for chatbot development. Some common techniques include:

* **Converting to lowercase:** Converting all words to lowercase.
* **Removing stop words:** Removing common words that do not add meaning to the sentence, such as "the", "is", and "of".
* **Lemmatizing:** Converting words to their base form.
* **Stemming:** Reducing words to their root form.
* **Punctuation removal:** Removing punctuation marks from the text.

The best normalization technique to use will depend on the specific chatbot application. For example, if you are developing a chatbot for customer service, you may want to remove punctuation marks from the text, as this can make it easier for the chatbot model to understand the user's query.

In [6]:

def remove\_diacritic(text):  
 return ''.join(char for char **in** unicodedata.normalize('NFD',text)  
 if unicodedata.category(char) !='Mn')

In [7]:

def preprocessing(text):  
   
 *#Case folding and removing extra whitespaces*

text=remove\_diacritic(text.lower().strip())  
   
 *#Ensuring punctuation marks to be treated as tokens*  
 text=re.sub(r"([?.!,¿])", r" \1 ", text)  
   
 *#Removing redundant spaces*  
 text= re.sub(r'[" "]+', " ", text)  
   
 *#Removing non alphabetic characters*  
 text=re.sub(r"[^a-zA-Z?.!,¿]+", " ", text)  
   
 text=text.strip()  
   
 *#Indicating the start and end of each sentence*  
 text='<start> ' + text + ' <end>'  
   
 return text

In [8]:

preprocessed\_questions=[preprocessing(sen) for sen **in** questions]  
 preprocessed\_answers=[preprocessing(sen) for sen **in** answers]  
  
 print(preprocessed\_questions[0])  
 print(preprocessed\_answers[0])

<start> hi , how are you doing ? <end>  
<start> i m fine . how about yourself ? <end>

## **Tokenization**

Tokenization is the process of splitting text into smaller units, called tokens. Tokens can be words, punctuation marks, or other symbols. Tokenization is an important step in chatbot development, as it allows the chatbot model to better understand the text data.

In [9]:

def tokenize(lang):  
 lang\_tokenizer = tf.keras.preprocessing.text.Tokenizer(  
 filters='')  
   
 *#build vocabulary on unique words*   
 lang\_tokenizer.fit\_on\_texts(lang)  
   
 return lang\_tokenizer

## **Word Embedding**

Word embedding is a technique for representing words as vectors of real numbers. Word embeddings are learned using machine learning algorithms, and they capture the semantic and syntactic relationships between words.

Word embeddings are useful for chatbot development because they allow chatbots to better understand the meaning of text. For example, if a user asks a chatbot "What is the meaning of life?", the chatbot can use word embeddings to understand that the words "meaning" and "life" are related, and that the user is asking a philosophical question.

In [10]:

def vectorization(lang\_tokenizer,lang):  
   
 *#word embedding for training the neural network*  
 tensor = lang\_tokenizer.texts\_to\_sequences(lang)  
  
 tensor = tf.keras.preprocessing.sequence.pad\_sequences(tensor,  
 padding='post')  
  
 return tensor