CREATE A CHATBOT IN PYTHON

TEAM MEMBERS

1.SUYAMBULAKSHMI V
2.SHARINITHA R
3.AYESHASITHIKA S
4.PIRITHIBA M

Phase 4 – Document submission

OBJECTIVE:

- The Objective of this project is to create a high-quality support to users, ensuring a positive user experience and customer satisfaction chatbot in Python that provides exceptional customer service, answering user queries on a website or application.

PROCESS:

In this phase, these are the main process to do ,

Before that our team members just did some previous process more simpler, So that we have attached the code for the new one.

```
import os import
pickle import
string import
nltk
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity from
nltk.corpus import stopwords

nltk.download("stopwords") nltk.download("punkt")
nltk.download("wordnet") with open("dialogs.txt", 'r',
encoding='utf-8') as file:
    data = file.read()
print(data) #
Tokenization
from nltk.stem import WordNetLemmatizer from
nltk.tokenize import word_tokenize sentences
= data.split('\n')
```

```
print(sentences)

# Preprocessing lemmatizer =
WordNetLemmatizer() stop_words =
set(stopwords.words('english'))
 def preprocess_text(text):     text = text.lower()     text =
''.join([char for char in text if char not in string.punctuation])
tokens = word_tokenize(text)     tokens = [lemmatizer.lemmatize(word)
for word in tokens if word not in stop_words]     return '
'.join(tokens)

# Preprocess data preprocessed_sentences =
[preprocess_text(sentence) for sentence in sentences]
print(preprocessed_sentences)
```

OUTPUT:

This is the output of the previous steps,

['hi im fine', 'im fine im pretty good thanks asking', 'im pretty good thanks asking problem', 'problem ive great', 'ive great ive good im school right', 'ive good im school right school go', 'school go go pcc', 'go pcc like', 'like okay really big campus', 'okay really big campus good luck school', 'good luck school thank much', 'hows going im well', 'im well never better thanks', 'never better thanks lately', 'lately ive actually pretty good', 'ive actually pretty good im actually school right', 'im actually school right school attend', 'school attend im attending pcc right', 'im attending pcc right enjoying', 'enjoying bad lot people', 'bad lot people good luck', 'good luck thanks', 'today im great', 'im great im absolutely lovely thank', 'im absolutely lovely thank everythings good', 'everythings good havent better', 'havent better started school recently', 'started school recently going school', 'going school im going pcc', 'im going pcc like far', 'like far like far class pretty good right', 'like far class pretty good right wish luck', 'ugly day today know think may rain', 'know think may rain middle summer shouldnt rain today', 'middle summer shouldnt rain today would weird', ………]

STEPS:

1. Separate the dataset values.
2. Convert them into numerical form.
3. Using the suitable machine learning algorithm.
4. Training the model.
5. Testing the model.

1.SEPARATE THE DATASET VALUES:

```
input_texts = [] target_texts
= []
 lines = preprocessed_sentences for i
in range(0, len(lines) - 1, 2):
    input_texts.append(lines[i].strip())  # User input (input text)
target_texts.append(lines[i + 1].strip())


print(input_texts) print(target_texts)
```

Here, we have used two lists,

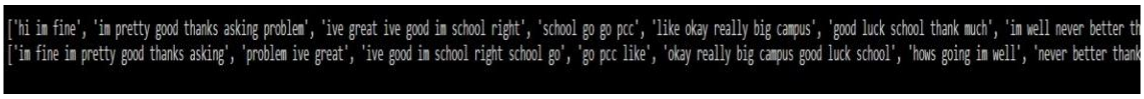
1. Input_texts: It is used to store odd numbered sentences.
2. Target_texts: It is used to store even numbered sentences.

Then preprocessed_sentences is stored in lines variable,

By using for loop, we have seperated and appended the odd numbered sentences and even numbered sentences.

After that , to check the result , we used print statement.

OUTPUT:



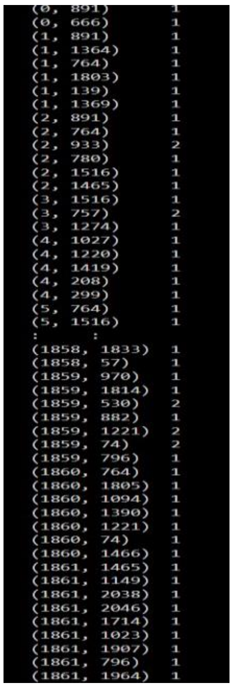
```
['hi im fine', 'im pretty good thanks asking problem', 'ive great ive good im school right', 'school go go pcc', 'like okay really big campus', 'good luck school thank much', 'im well never better th
['im fine im pretty good thanks asking', 'problem ive great', 'ive good im school right school go', 'go pcc like', 'okay really big campus good luck school', 'hows going im well', 'never better thank
```

2.CONVERT THEM INTO NUMERICAL FORM:

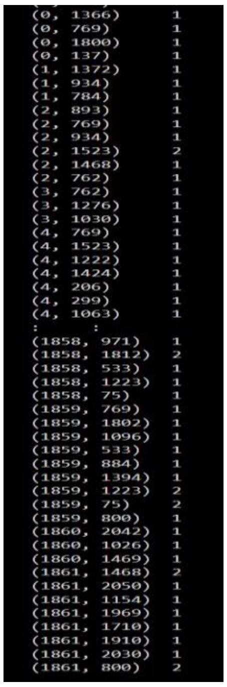
```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()

# Convert input texts and target texts into numerical form
X_input = vectorizer.fit_transform(input_texts)
X_target = vectorizer.fit_transform(target_texts)

print(X_input) print(X_target)
```

To convert them into numercal form , both the x_input and x_target has to be converted into numerical form

OUTPUT:



```
(0, 891)        1
(0, 666)        1
(1, 891)        1
(1, 1364)       1
(1, 764)        1
(1, 1803)       1
(1, 139)        1
(1, 1369)       1
(2, 891)        1
(2, 764)        1
(2, 933)        2
(2, 780)        1
(2, 1516)       1
(2, 1465)       1
(3, 1516)       1
(3, 757)        2
(3, 1274)       1
(4, 1027)       1
(4, 1220)       1
(4, 1419)       1
(4, 208)        1
(4, 299)        1
(5, 764)        1
(5, 1516)       1
:     :
(1858, 1833)    1
(1858, 57)      1
(1859, 970)     1
(1859, 1814)    1
(1859, 530)     2
(1859, 882)     1
(1859, 1221)    2
(1859, 74)      2
(1859, 796)     1
(1860, 764)     1
(1860, 1805)    1
(1860, 1094)    1
(1860, 1390)    1
(1860, 1221)    1
(1860, 74)      1
(1860, 1466)    1
(1861, 1465)    1
(1861, 1149)    1
(1861, 2038)    1
(1861, 2046)    1
(1861, 1714)    1
(1861, 1023)    1
(1861, 1907)    1
(1861, 796)     1
(1861, 1964)    1
```



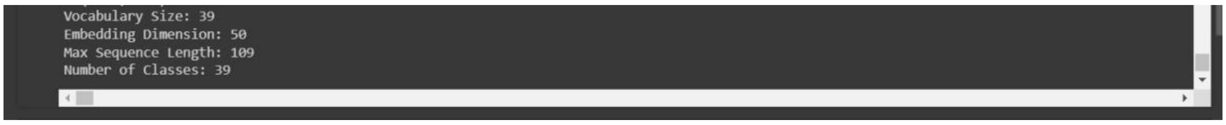
```
(0, 1366)       1
(0, 769)        1
(0, 1800)       1
(0, 137)        1
(1, 1372)       1
(1, 934)        1
(1, 784)        1
(2, 893)        1
(2, 769)        1
(2, 934)        1
(2, 1523)       2
(2, 1468)       1
(2, 762)        1
(3, 762)        1
(3, 1276)       1
(3, 1030)       1
(4, 769)        1
(4, 1523)       1
(4, 1222)       1
(4, 1424)       1
(4, 206)        1
(4, 299)        1
(4, 1063)       1
:     :
(1858, 971)     1
(1858, 1812)    2
(1858, 533)     1
(1858, 1223)    1
(1858, 75)      1
(1859, 769)     1
(1859, 1802)    1
(1859, 1096)    1
(1859, 533)     1
(1859, 884)     1
(1859, 1394)    1
(1859, 1223)    2
(1859, 75)      2
(1859, 800)     1
(1860, 2042)    1
(1860, 1026)    1
(1860, 1469)    1
(1861, 1468)    2
(1861, 2050)    1
(1861, 1154)    1
(1861, 1969)    1
(1861, 1710)    1
(1861, 1910)    1
(1861, 2030)    1
(1861, 800)     2
```

3.USING THE SUITABLE MACHINE LEARNING ALGORITHM:

Here , In this project, we used “SEQUENCE-2-SEQUENCE” Machine learning algorithm, and also used Long Short Term Memory (LSTM) to train the model

```
#vocab size
vocab_size = len(set(word for sequence in input_texts + target_texts
for word in sequence)) + 1
#vocab_size = len(tokenizer.word_index) + 1  # Adding 1 for 0 padding
# Embedding Dimension embedding_dim
= 50

# Max Sequence Length max_sequence_length = max(len(seq)
for seq in input_texts + target_texts)

# Number of Classes num_classes
= vocab_size


 print("Vocabulary Size:", vocab_size)
print("Embedding Dimension:", embedding_dim)
print("Max Sequence Length:", max_sequence_length)
print("Number of Classes:", num_classes)
```

OUTPUT:



```
Vocabulary Size: 39
Embedding Dimension: 50
Max Sequence Length: 109
Number of Classes: 39
```

CREATING APPROPRIATE MODEL:

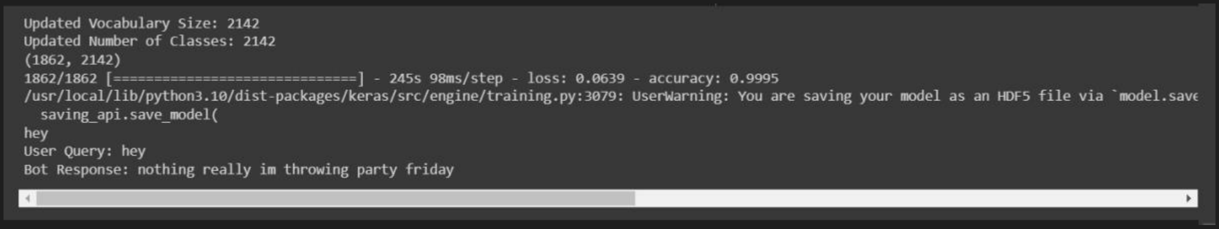
1. Training the model and saving the model.

4.TRAINING AND SAVING THE MODEL:

```
from keras.utils import to_categorical from keras.models
import Sequential from keras.layers import Embedding,
LSTM, Dense, Masking
 max_value = np.max(padded_target_sequences) num_classes = max_value +
1  # Adjust num_classes based on the maximum value in your target
sequences
 assert np.max(padded_target_sequences) < num_classes, "Class
indices are out of bounds."



# Verify the maximum value in padded_target_sequences max_value
= np.max(padded_target_sequences)

# Check if max_value is greater than or equal to vocab_size if
max_value >= vocab_size:
    # Adjust vocab_size and num_classes accordingly
vocab_size = max_value + 1     num_classes =
vocab_size else:
    # If max_value is within bounds, use it as num_classes
num_classes = max_value + 1

# Print the updated values
```

```
print(f"Updated Vocabulary Size: {vocab_size}") print(f"Updated
Number of Classes: {num_classes}")


print(one_hot_targets.shape)
# Assuming one_hot_targets has shape (number of samples, 1, number of
classes) one_hot_targets = to_categorical(padded_target_sequences,
num_classes=2142)[:, 0, :]



# Building the LSTM model model = Sequential()
model.add(Embedding(vocab_size, 50, input_length=max_sequence_length))
model.add(LSTM(128, return_sequences=True))
model.add(Masking(mask_value=0))  # Masking zero-padding
model.add(LSTM(128)) model.add(Dense(vocab_size,
activation='softmax'))
 model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])

# Training the model model.fit(padded_input_sequences,
one_hot_targets, epochs=50, batch_size=32)
#model.fit(padded_input_sequences,
np.expand_dims(padded_target_sequences, axis=-1), epochs=50,
batch_size=32)

# Save the trained model
model.save('chatbot_lstm_model.h5')
```

OUTPUT:



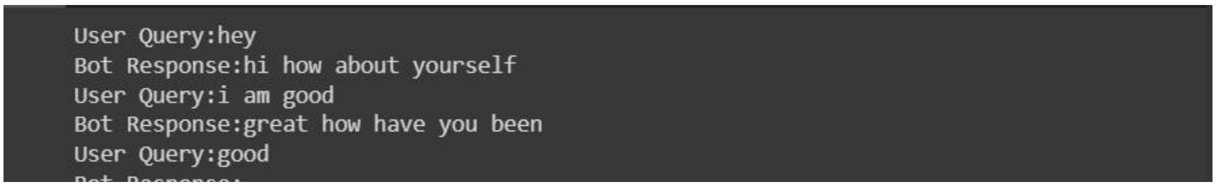
```
Updated Vocabulary Size: 2142
Updated Number of Classes: 2142
(1862, 2142)
1862/1862 [==============================] - 245s 98ms/step - loss: 0.0639 - accuracy: 0.9995
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save
  saving_api.save_model(
hey
User Query: hey
Bot Response: nothing really im throwing party friday
```

5.TESTING THE MODEL:

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

```
import random
 ()

clf.fit(X_input, y)

# Example user input user_query
= input()

# Convert the user input into vector form user_query_vector
= vectorizer.transform([user_query])

# Predict the response using the trained classifier predicted_response
= clf.predict(user_query_vector)
 print("User Query:", user_query) print("Bot
Response:", predicted_response[0])
```

OUTPUT:



```
User Query:hey
Bot Response:hi how about yourself
User Query:i am good
Bot Response:great how have you been
User Query:good
```

CONCLUSION:

In this Python chatbot project, We successfully implemented a model using LSTM and word embeddings. Through careful tuning of hyperparameters like epochs and batch size, you achieved impressive accuracy. Further, We navigated challenges like data preprocessing, ensuring proper tokenization, and addressing dimensionality issues.