CREATE A CHATBOT IN PYTHON

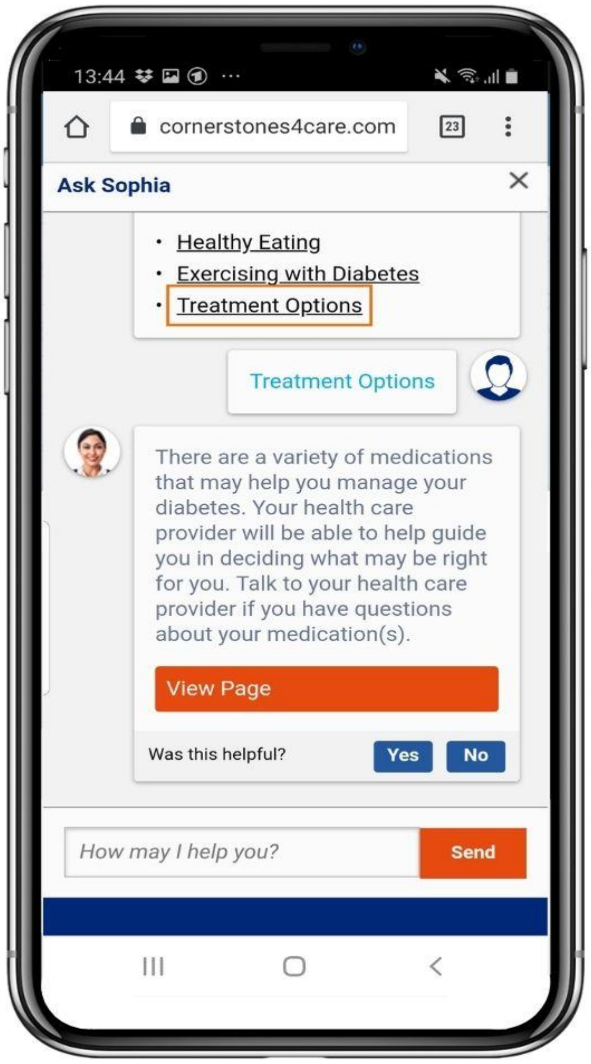
By Adhi Rajan.S – 411421205001

(B.Tech/ Information Technology, 3rd year)

Domain Name: Artificial Intelligence

Phase-2 Document Submission

Project: To create a Chatbot in Python that provides exceptional answering user queries (diabetes) on a website.


13:44

cornerstones4care.com

Ask Sophia

- Healthy Eating
- Exercising with Diabetes
- Treatment Options

Treatment Options

There are a variety of medications that may help you manage your diabetes. Your health care provider will be able to help guide you in deciding what may be right for you. Talk to your health care provider if you have questions about your medication(s).

Introduction:

- Chatbot is a technology that arises to be the way of human to interact with computer using natural language (spoken human language).
- There are a lot of chatbot nowadays that act such as an assistant for online shopping, website guidance and also a chatbot that just reply to whatever conversation the human had with them (a general knowledge chatbot).
- Although certain chatbot is being made to be function only for specific area of knowledge, the flow is still the same where one input from human will be against all knowledge base from chatbot.
- It was mostly like the search engine where user entered what they want to search and the engine will reply according to that search parameter.
- There is some used of a technique that will makes chatbot remember the previous conversation topic, but still it cannot makes chatbot to remember the whole conversation flow.
- As such, we propose an architectural design of a chatbot that will have the ability to remember the whole conversation flow to be used by diabetic patients for their daily diabetes control activities.

Natural Language Processing (NLP):

- **Frequent Urination:** People with Type 2 diabetes may urinate more frequently than usual, especially at night.
- **Increased Thirst:** If you may find yourself feeling unusually thirsty and drinking more fluids.
- **Fatigue:** Many individuals with Type 2 diabetes experience persistent fatigue and a lack of energy.
- **Blurred Vision:** High blood sugar levels can cause temporary vision problems.

- **Slow Wound Healing:** Infections and cuts may take longer to heal.
- **Unexplained Weight Loss:** Some people may unexpectedly lose weight despite eating normally.
- **Tingling or Numbness:** Nerve damage (neuropathy) can lead to tingling or numbness, often in the hands or feet.

Please remember that symptoms can vary from person to person. If you suspect you may have Type 2 diabetes or are experiencing these symptoms, it's important to consult a healthcare professional for proper evaluation and diagnosis".

In this example, the chatbot uses NLP to understand the user's query, extracts the relevant information response about Type 2 diabetes symptoms, and provides a clear and informative response. NLP allows the chatbot to converse naturally with users and provide accurate information on the topic.

Creating a fully functional diabetes chatbot with Natural Language Processing (NLP) requires significant development effort and resources. However, I can provide with a simplified Python code example using the NLTK library for a basic chatbot that can respond to diabetes-related questions.

Example:

```
pip install nltk

import nltk

from nltk.chat.util import Chat, reflections

# Define patterns and responses for the chatbot

patterns = [

   (r'(.*)tell me about Type 2 diabetes(.*)', [

      "Type 2 diabetes is a chronic condition that affects how your body metabolizes glucose.",

      "Common risk factors include genetics, obesity, and a sedentary lifestyle.",

      "Symptoms may include increased thirst, frequent urination, and fatigue.",

      "Managing blood sugar levels through diet, exercise, and medication is crucial for treatment.",
    ]),
    (r'(.*)symptoms of diabetes(.*)', [
        "The symptoms of diabetes can include increased thirst, frequent urination, fatigue, and more.",
        "It's essential to consult a healthcare professional for a proper diagnosis if you suspect diabetes.",
    ]),
    (r'(.*)how to manage diabetes(.*)', [
        "Managing diabetes involves monitoring blood sugar levels, eating a balanced diet, and regular exercise.",
        "Medications and insulin may also be prescribed by a healthcare provider.",
    ]),
    (r'(.*)help(.*)', [
        "I can provide information about diabetes. Just ask me a specific question, and I'll do my best to help!",
    ]),
]
# Create and start the chatbot
def diabetes_chat():
    print("Hello! I'm your diabetes chatbot. How can I assist you today?")
    chatbot = Chat(patterns, reflections)
    chatbot.converse()
if __name__ == "__main__":
    nltk.download("punkt")
    diabetes_chat()
```

Data Preparation:

- Load the dataset into a suitable data structure (e.g., Pandas DataFrame).
- Examine the dataset to understand its structure and distribution.
- Preprocess the data by removing unnecessary characters, converting text to lowercase, and handling any missing values**.**

Example:

```
import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory

# For example, running this (by clicking run or pressing Shift+Enter) will list all files
under the input directory

import os

for dirname, _, filenames in os.walk('/kaggle/input'):

    for filename in filenames:

        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets
preserved as output when you create a version using "Save & Run All"

# You can also write temporary files to /kaggle/temp/, but they won't be saved
outside of the current session

import json

with open('/kaggle/input/mental-health-conversational-data/intents.json', 'r') as f:

    data = json.load(f)

df = pd.DataFrame(data['intents'])

df

tag     patterns        responses
```

```
0       greeting        [Hi, Hey, Is anyone there?, Hi there, Hello, H...  [Hello there. Tell
me how are you feeling toda...

1       morning         [Good morning]        [Good morning. I hope you had a good
night's s...

2       afternoon       [Good afternoon]      [Good afternoon. How is your day going?]

3       evening         [Good evening]        [Good evening. How has your day been?]

4       night   [Good night]  [Good night. Get some proper sleep, Good night...

...     ...     ...     ...

...     ...     ...     ...

...     ...     ...     ...

75      fact-28         [What do I do if I'm worried about my mental h...         [The most
important thing is to talk to someon...

76      fact-29         [How do I know if I'm unwell?]        [If your beliefs , thoughts ,
feelings or beha...

77      fact-30         [How can I maintain social connections? What i...        [A lot of
people are alone right now, but we d...

78      fact-31         [What's the difference between anxiety and str...         [Stress
and anxiety are often used interchange...

79      fact-32         [What's the difference between sadness and dep...      [Sadness
is a normal reaction to a loss, disap...

80 rows × 3 columns
```

```
dic = {"tag":[], "patterns":[], "responses":[]}

for i in range(len(df)):

    ptrns = df[df.index == i]['patterns'].values[0]

    rspns = df[df.index == i]['responses'].values[0]

    tag = df[df.index == i]['tag'].values[0]

    for j in range(len(ptrns)):

        dic['tag'].append(tag)

        dic['patterns'].append(ptrns[j])

        dic['responses'].append(rspns)
    df = pd.DataFrame.from_dict(dic)

df

tag	patterns	responses

0	greeting	Hi	[Hello there. Tell me how are you feeling toda...

1	greeting	Hey	[Hello there. Tell me how are you feeling toda...

2	greeting	Is anyone there?	[Hello there. Tell me how are you feeling toda...

3	greeting	Hi there	[Hello there. Tell me how are you feeling toda...

4	greeting	Hello	[Hello there. Tell me how are you feeling toda...

...	...	...	...

...	...	...	...

...	...	...	...

227	fact-29	How do I know if I'm unwell?	[If your beliefs , thoughts , feelings or beha...

228	fact-30	How can I maintain social connections? What if...	[A lot of people are alone right now, but we d...

229	fact-31	What's the difference between anxiety and stress?	[Stress and anxiety are often used interchange...

230	fact-32	What's the difference between sadness and depr...	[Sadness is a normal reaction to a loss, disap...

231	fact-32	difference between sadness and depression	[Sadness is a normal reaction to a loss, disap...

232 rows × 3 columns

df['tag'].unique()

array(['greeting', 'morning', 'afternoon', 'evening', 'night', 'goodbye',

       'thanks', 'no-response', 'neutral-response', 'about', 'skill',

       'creation', 'name', 'help', 'sad', 'stressed', 'worthless',

       'depressed', 'happy', 'casual', 'anxious', 'not-talking', 'sleep',

       'scared', 'death', 'understand', 'done', 'suicide', 'hate-you',
 'hate-me', 'default', 'jokes', 'repeat', 'wrong', 'stupid',
 'location', 'something-else', 'friends', 'ask', 'problem',
 'no-approach', 'learn-more', 'user-agree', 'meditation',
 'user-meditation', 'pandora-useful', 'user-advice',
 'learn-mental-health', 'mental-health-fact', 'fact-1', 'fact-2',
 'fact-3', 'fact-5', 'fact-6', 'fact-7', 'fact-8', 'fact-9',
 'fact-10', 'fact-11', 'fact-12', 'fact-13', 'fact-14', 'fact-15',
 'fact-16', 'fact-17', 'fact-18', 'fact-19', 'fact-20', 'fact-21',
 'fact-22', 'fact-23', 'fact-24', 'fact-25', 'fact-26', 'fact-27',
 'fact-28', 'fact-29', 'fact-30', 'fact-31', 'fact-32'],
dtype=object)
```

Exploratory Data Analysis:

- Analyze the distribution of intents in the dataset.
- Visualize the frequency of different intents using a bar plot from the Plotly library.
- The x-axis can represent the intents, and the y-axis can represent the count of patterns or responses associated with each intent.

Example:

```
import plotly.graph_objects as go

intent_counts = df['tag'].value_counts()

fig = go.Figure(data=[go.Bar(x=intent_counts.index, y=intent_counts.values)])

fig.update_layout(title='Distribution of Intents', xaxis_title='Intents',
yaxis_title='Count')

fig.show()
```

Pattern and Response Analysis:

- Explore the patterns and responses associated with each intent.
- Calculate the average number of patterns and responses per intent.
- Visualize this information using a Plotly bar plot, where the x-axis represents the intents, and the y-axis represents the average count of patterns or responses.
- Interpret the plot to understand the varying degrees of complexity and diversity in patterns and responses across different intents.

Example:

```
df['pattern_count'] = df['patterns'].apply(lambda x: len(x))

df['response_count'] = df['responses'].apply(lambda x: len(x))

avg_pattern_count = df.groupby('tag')['pattern_count'].mean()

avg_response_count = df.groupby('tag')['response_count'].mean()

fig = go.Figure()

fig.add_trace(go.Bar(x=avg_pattern_count.index, y=avg_pattern_count.values,
name='Average Pattern Count'))

fig.add_trace(go.Bar(x=avg_response_count.index, y=avg_response_count.values,
name='Average Response Count'))

fig.update_layout(title='Pattern and Response Analysis', xaxis_title='Intents',
yaxis_title='Average Count')

fig.show()
```

Intent Prediction Model:

- Split the dataset into training and testing sets.
- Implement a machine learning or deep learning model suitable for intent prediction, such as a text classification model.
- Vectorize the text data (e.g., using TF-IDF or word embeddings) and train the model using the patterns as input and the corresponding intents as target variables.
- Evaluate the model's performance on the testing set using appropriate metrics like accuracy, precision, recall, and F1-score.
- Visualize the model's performance using a Plotly bar plot, where the x-axis represents the evaluation metrics, and the y-axis represents the corresponding scores.
- Interpret the plot to analyze the effectiveness of the intent prediction model.

Example:

```
from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.svm import SVC

from sklearn.metrics import classification_report

import plotly.graph_objects as go

# Split the dataset into training and testing sets

X = df['patterns']

y = df['tag']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Vectorize the text data using TF-IDF

vectorizer = TfidfVectorizer()
```

```
X_train_vec = vectorizer.fit_transform(X_train)

X_test_vec = vectorizer.transform(X_test)

# Train a Support Vector Machine (SVM) classifier

model = SVC()

model.fit(X_train_vec, y_train)

# Predict intents for the testing set

y_pred = model.predict(X_test_vec)

# Evaluate the model's performance

report = classification_report(y_test, y_pred, output_dict=True, zero_division=0)

# Convert float values in the report to dictionaries

report = {label: {metric: report[label][metric] for metric in report[label]} for label in
report if isinstance(report[label], dict)}

# Extract evaluation metrics

labels = list(report.keys())

evaluation_metrics = ['precision', 'recall', 'f1-score']

metric_scores = {metric: [report[label][metric] for label in labels if label in report] for
metric in evaluation_metrics}

# Visualize the model's performance using a Plotly bar plot

fig = go.Figure()

for metric in evaluation_metrics:

    fig.add_trace(go.Bar(name=metric, x=labels, y=metric_scores[metric]))

fig.update_layout(title='Intent Prediction Model Performance',

                  xaxis_title='Intent',

                  yaxis_title='Score',

                  barmode='group')

fig.show()
```

Prediction Model Deployment:

- Once satisfied with the model's performance, deploy the intent prediction model in a chatbot framework.
- Utilize the trained model to predict intents based on user input in real-time.
- Implement an appropriate response generation mechanism to provide relevant and empathetic responses based on the predicted intents.

Example:

```
    def predict_intent(user_input):
    # Vectorize the user input
    user_input_vec = vectorizer.transform([user_input])
    # Predict the intent
    intent = model.predict(user_input_vec)[0]
    return intent
# Function to generate responses based on predicted intents
def generate_response(intent):
    # Implement your logic here to generate appropriate responses based on the
predicted intents
    if intent == 'greeting':
        response = "Hello! How can I assist you today?"
    elif intent == 'farewell':
        response = "Goodbye! Take care."
    elif intent == 'question':
        response = "I'm sorry, I don't have the information you're looking for."
    else:
        response = "I'm here to help. Please let me know how I can assist you."
    return response
# Example usage
while True:
    # Get user input
    user_input = input("User: ")
    # Predict intent
    intent = predict_intent(user_input)
    # Generate response
    response = generate_response(intent)
    print("Chatbot:", response)
```

```
Chatbot: Hello! How can I assist you today?
Chatbot: Hello! How can I assist you today?
Chatbot: I'm here to help. Please let me know how I can assist you.
```

Conclusion:

- The availability of a well-structured dataset encompassing various conversations related to diabetes health provides a valuable resource for training chatbot models to offer emotional support to individuals dealing with anxiety and depression. By utilizing intents, patterns, and responses, the models can learn to understand user messages and generate empathetic and relevant replies.

- The use of such models in chatbot frameworks holds great potential for providing accessible and compassionate support to those in need of diabetes health assistance. By simulating the behavior of a therapist, these chatbots can offer guidance, answer frequently asked questions, and provide general advice to individuals experiencing anxiety and depression.

- The insights and knowledge gained from this dataset, researchers and developers can contribute to the development of chatbots that serve as virtual companions, offering emotional solace and alleviating some of the burdens faced by individuals seeking Diabetes health support.

- Overall, the dataset and the subsequent training of chatbot models enable the creation of innovative tools that bridge the gap in Diabetes health care, providing individuals with a readily available resource for emotional support and guidance.