CHATBOT USING PYTHON

RAGHUL S

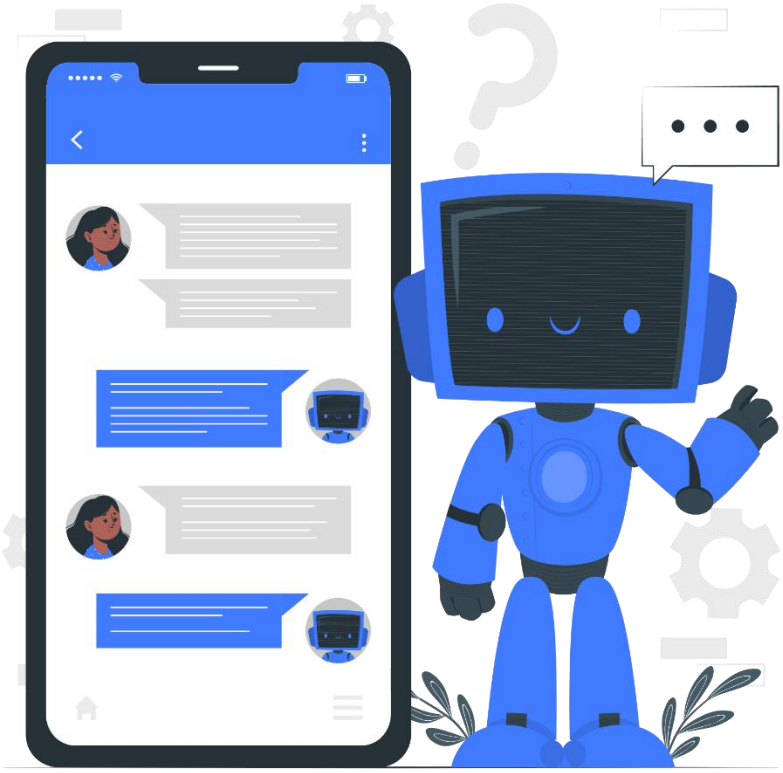
211121205023

Phase 3 submission document

Project Title: Create a chatbot in Python

Phase 3: *Development Part 1*

Topic: *Start building the chatbot by loading and pre-processing the dataset and preparing the environment and implementing basic user interactions.*



CHATBOTS

INTRODUCTION:

- Imagine having an employee on your team who is available 24/7, never complains, and will do all the repetitive customer service tasks that your other team members hate. They cost a fraction of your average employee's salary.

- This unicorn of a worker exists, just not in the traditional human sense. Chatbots are many businesses' next competitive edge. The multiple benefits of chatbots give them a ton of bang for their buck.

- Businesses commonly use chatbots to help customers with customer service, inquiries, and sales. But that's just scratching the surface of how you can use chatbots for business.

- Chatbots can be programmed to respond to certain keywords in a specific way. Or, you can use machine learning to train your chatbots to respond organically.

- There are two main camps for chatbots:

    1. Smart chatbots are AI-driven.

    2. Simple chatbots are rule-based.

- Simple chatbots work like a flowchart. If someone asks them X, they respond with Y. You'll program these bots in the beginning to do your bidding. Then, so long as customers are clear and straightforward in their questions, they'll get to where they need to go.

- Smart chatbots, however, use machine learning to understand the context and intent behind questions or queries. These bots generate answers using natural language processing.

Given data set:

```
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.
i've been good. i'm in school right now.        what school do you go to?
what school do you go to?       i go to pcc.
i go to pcc.    do you like it there?
do you like it there?   it's okay. it's a really big campus.
it's okay. it's a really big campus.    good luck with school.
good luck with school.  thank you very much.
how's it going? i'm doing well. how about you?
i'm doing well. how about you?  never better, thanks.
never better, thanks.   so how have you been lately?
so how have you been lately?    i've actually been pretty good. you?
i've actually been pretty good. you?    i'm actually in school right now.
i'm actually in school right now.       which school do you attend?
which school do you attend?     i'm attending pcc right now.
i'm attending pcc right now.    are you enjoying it there?
are you enjoying it there?      it's not bad. there are a lot of people there.
it's not bad. there are a lot of people there.  good luck with that.
good luck with that.    thanks.
```

It consists of two columns: question \t answer \n . Suitable for simple chatbots. Contains 3725 items.

- This dataset contains predefined questions and responses for a straightforward chatbot. If we want to expand the use of chatbots, we can use transformers for GPT-3 to enhance their capabilities.
- Generative Pre-trained Transformer 3 (GPT-3) is a large language model released by OpenAI in 2020. Like its predecessor GPT-2, it is a decoder-only transformer model of deep neural network, which uses attention in place of previous recurrence- and convolution-based architectures.
- It uses a 2048-tokens-long context and then-unprecedented size of 175 billion parameters, requiring 800GB to store.

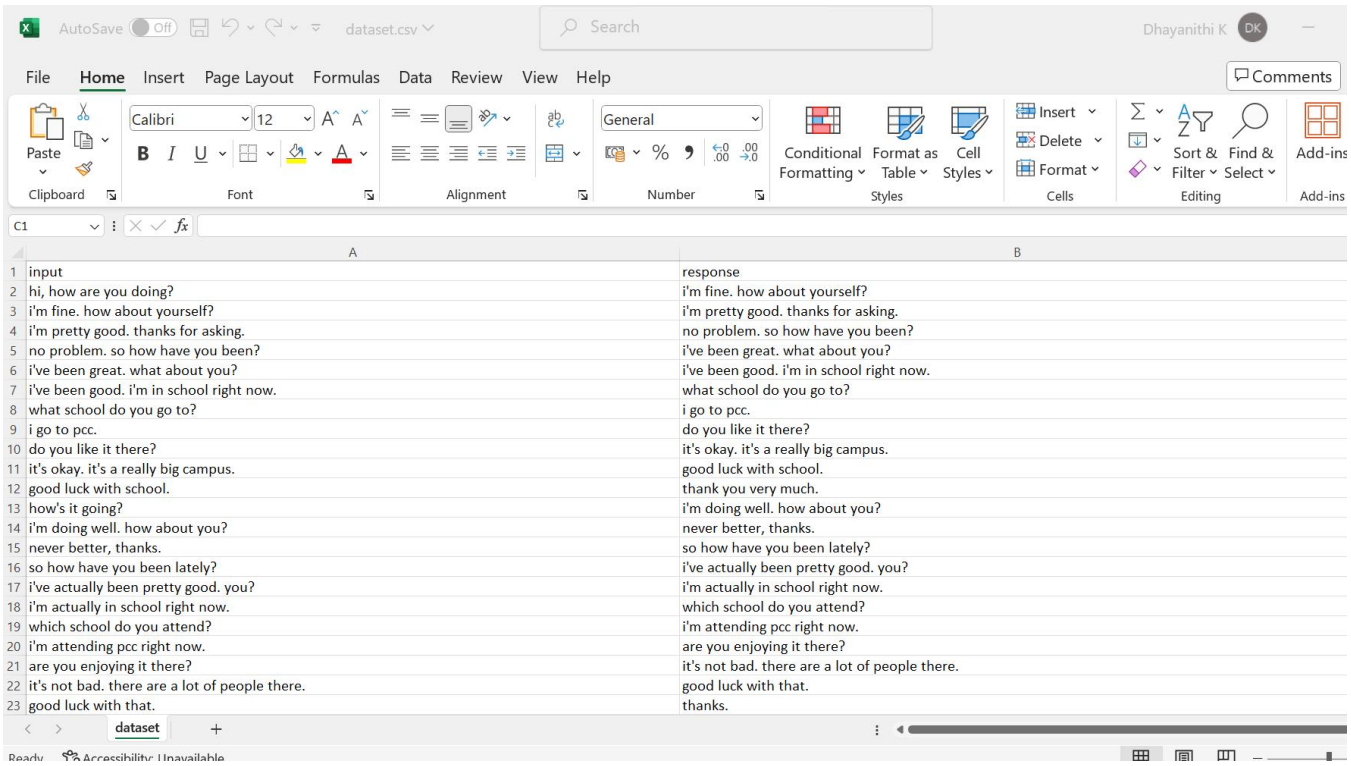
Converting dataset into CSV file:

Program:

```
#importing libraries
import csv

# Define the input and output file paths

input_file_path = r'C:\\Users\\dhaya\\Downloads\\archive\\dialogs.txt'
output_file_path = 'C:\\Users\\dhaya\\OneDrive\\Desktop\\dataset.csv'

# Open the input text file for reading and CSV file for writing

with open(input_file_path, 'r', encoding='utf-8') as input_file,
open(output_file_path, 'w', newline='', encoding='utf-8') as output_file:
    csv_writer = csv.writer(output_file)
    csv_writer.writerow(['input', 'response'])  # Write the header row

    for line in input_file:
        line = line.strip()
        if line:
 # Check if the line is not empty
 # Split the line into input and response using a separator (e.g., '\t' for tab)
            input_text, response_text = line.split('\t')

# Write the input and response to the CSV file
            csv_writer.writerow([input_text, response_text])

print("Conversion completed. CSV file saved at {output_file_path}")
```

Output: Conversion completed. CSV file saved at C:\\Users\\dhaya\\OneDrive\\Desktop\\dataset.csv

Hence required csv file is created using above python code.



	A	B
1	input	response
2	hi, how are you doing?	i'm fine. how about yourself?
3	i'm fine. how about yourself?	i'm pretty good. thanks for asking.
4	i'm pretty good. thanks for asking.	no problem. so how have you been?
5	no problem. so how have you been?	i've been great. what about you?
6	i've been great. what about you?	i've been good. i'm in school right now.
7	i've been good. i'm in school right now.	what school do you go to?
8	what school do you go to?	i go to pcc.
9	i go to pcc.	do you like it there?
10	do you like it there?	it's okay. it's a really big campus.
11	it's okay. it's a really big campus.	good luck with school.
12	good luck with school.	thank you very much.
13	how's it going?	i'm doing well. how about you?
14	i'm doing well. how about you?	never better, thanks.
15	never better, thanks.	so how have you been lately?
16	so how have you been lately?	i've actually been pretty good. you?
17	i've actually been pretty good. you?	i'm actually in school right now.
18	i'm actually in school right now.	which school do you attend?
19	which school do you attend?	i'm attending pcc right now.
20	i'm attending pcc right now.	are you enjoying it there?
21	are you enjoying it there?	it's not bad. there are a lot of people there.
22	it's not bad. there are a lot of people there.	good luck with that.
23	good luck with that.	thanks.

Loading and Pre-Processing the Dataset:

Data Cleansing:

- Data cleansing, also known as data cleaning or data scrubbing, is the process of identifying and correcting errors or inconsistencies in a dataset to improve its quality and reliability. Like

1. Remove HTML tags, duplicates, special characters, and noise from the text.
2. Convert text to lowercase to ensure uniformity.
3. Install the required libraries if you haven't already:

   pip install pandas beautifulsoup4

Program:

```
#importing libraries
import pandas as pd
from bs4 import BeautifulSoup
import re

# Function to clean and preprocess text
def clean_text(text):
    # Remove HTML tags
    text = BeautifulSoup(text, "html.parser").get_text()

    # Convert to lowercase
    text = text.lower()

    # Remove special characters and noise
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
    return text

# Load the CSV file
input_file = 'your_input_file.csv'
output_file = 'cleaned_output_file.csv'
df = pd.read_csv(input_file)

# Clean and preprocess the text in the 'text_column' of the CSV
df['text_column'] = df['text_column'].apply(clean_text)

# Remove duplicates
df.drop_duplicates(subset='text_column', keep='first', inplace=True)

# Save the cleaned data to a new CSV file
df.to_csv(output_file, index=False)
```

Hence required data cleansed csv file is created and saved using above python code.

Tokenization:

- To format a dataset into input and target pairs that are compatible with GPT-3, you need to prepare your data in a specific format called "prompt-style completion." GPT-3 is a language model that generates text based on a prompt or input text, and it's often used for tasks like text generation, completion, and language understanding.

Here's how to format your dataset into input and target pairs compatible with GPT-3:

1. Create Input Prompts: The input prompt is the text that you provide to GPT-3 to generate a continuation or response. You can frame this prompt as a statement, question, or any text that sets the context for the desired output.
2. Specify the Target: The target is what you want GPT-3 to generate or complete. This is the text that you expect GPT-3 to produce as an output based on the given input prompt.
3. Combine Input and Target: Combine the input prompt and target into a single string, with a clear delimiter to separate them. A common practice is to use a special token like "<|endoftext|>" to separate the input and target.

Program:

```
import csv

# Define the CSV file path, input column name, and target column name

csv_file = 'dataset.csv'

input_column_name = 'input'

target_column_name = 'response'
```

```
# Open the CSV file and create a new text file for formatted data
formatted_file = 'formatted_data.txt'
with open(csv_file, 'r', newline='') as csvfile, open(formatted_file, 'w') as outfile:
    csvreader = csv.DictReader(csvfile)
    for row in csvreader:
        input_text = row[input]
        target_text = row[response]

        # Format the data as input-target pairs separated by '<|endoftext|>'
        formatted_pair = f"{input_text}<|endoftext|>\n{target_text}\n"

        # Write the formatted pair to the output text file
        outfile.write(formatted_pair)

print("Formatted data has been saved to", formatted_file)
```

This code reads the CSV file, formats the data into pairs with the <|endoftext|> token as a separator, and saves the pairs to the formatted_data.txt text file. You can then use this formatted text file with GPT-3 for text generation or completion tasks.

GPT-3 Integration:

Install the Transformers library:

pip install transformers

Using GPT-3 with Transformers

Program:

```
from transformers import GPT2LMHeadModel, GPT2Tokenizer
  # Load GPT-2 model and tokenizer
  model = GPT2LMHeadModel.from_pretrained("gpt2")
  tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
  # Read input and response pairs from a text file
  with open("formatted_data.txt", "r") as file:
     lines = file.readlines()
  # Process each line to separate input and response
  for line in lines:
     # Split the line into input and response using a delimiter (e.g., '\t' for tab-separated)
     input_text, response_text = line.strip().split('\t')
     # Encode the user's input
     input_ids = tokenizer.encode(input_text, return_tensors="pt")
     # Generate a response
     output = model.generate(input_ids, max_length=50, num_return_sequences=1)
     generated_response = tokenizer.decode(output[0],
```

```
skip_special_tokens=True)
    # Print or use the input and response as needed
    print("User's message:", input_text)
    print("Response:", response_text)
print("Generated response:", generated_response)
```

Web App Development with Flask

- Flask is a micro web framework for Python that is widely used for web app development. It's known for its simplicity and flexibility, making it an excellent choice for developing web applications of various sizes and complexities.

Installation:

First, make sure you have Python installed. You can then install Flask using pip:

pip install Flask

Project Structure:

Organize your project by creating a directory structure. A basic structure might look like this:

```
my_flask_app/
    ├── app.py
    ├── templates/
    │   └── index.html
    └── static/
     └── style.css
```

App.py

Program;

Import necessary libraries:

```
import csv
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from flask import Flask, render_template, request


app = Flask(Chatbot)
# Define a function to load questions and answers from a CSV file
def load_qa_data(filename):
    qa_data = []
    with open(filename, newline='', encoding='utf-8') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            if len(row) == 2:
                question, answer = row
                qa_data.append((clean_text(question), clean_text(answer)))
    return qa_data
```

```
# Data cleansing function
def clean_text(text):
    # Convert to lowercase and remove extra spaces
    text = text.lower().strip()

    # Remove punctuation and special characters
    text = re.sub(r'[^\w\s]', '', text)

    # Tokenize the text and remove stopwords
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word not in
stopwords.words('english')]

    # Rejoin the tokens into a cleaned text
    cleaned_text = ' '.join(tokens)
    return cleaned_text

# Load questions and answers from the CSV file
qa_data = load_qa_data('dataset.csv')
```

```
@app.route('/')
def index():
    return render_template('index.html')


@app.route('/chat', methods=['POST'])
def chat():
    user_input = request.form['user_input']
    user_input = clean_text(user_input)
    response = "I'm sorry, I don't have an answer to that question."
    for question, resp in qa_data:
        if user_input == question:
            response = resp
            break
    return response


if __name__ == "__main__":
 app.run(debug=True)
```

- In this code, we import Flask and create a Flask app. We define two routes: '/' for the main page and '/chat' to handle user interactions.
- Create an HTML template for the user interface. Create a folder named templates in your project directory, and inside it, create an HTML file named index.html. Here's a simple example of what it could look like:

Conclusion:

- In the quest to build a chatbot using python, we have embarked on a critical journey that begins with loading and preprocessing the dataset. We have traversed through essential steps, starting with importing the necessary libraries to facilitatedata manipulation and analysis.
- Data preprocessing emerged as a pivotal aspect of this process. It involves cleaning, transforming, and refining the dataset to ensurethat it aligns with the requirements of machine learning algorithms.
- With these foundational steps completed, our dataset is now primed for the subsequent stages of building and training a chatbot.