Mental Health FAQ chatbot documentation

Done By : Chiraz BOUDERBALI and Meriem SELAMA

Table of content

Mental health FAQ chatbot

This project was done by:

- Chiraz Bouderbali
- Meriem Selama

1. Project Description

This project is a chatbot that answers different frequently asked questions about mental health, including depression, bipolar disorder…

2. Tools and libraries:

- Python: as the main programming language.
- Google Colab: a web-based notebook.
- Pandas: a Python library that is dedicated to data analysis, it helps to read the dataset within Colab and creating dataframes.
- Scikitlearn: a python library that is used for data prediction, in our case, it is used for vectorization.
- Spacy: a python module that is used for NLP (natural language processing).
- Re: Regular expression: a python module that is used for text cleaning.
- Cosine similarity: to calculate the similarity of two vectors of scores.
- Sentence transformer: to help enhancing the chatbot, using a pre-leaned model: SBERT
- Streamlit: to create a simple interface where the user can interact with the chatbot.
- Pyngrok: an ngrok python library to easily deploy the chatbot by generating a link to access the chatbot on browsers.

It aims to help the users find answers to their different questions and have a knowledge about the mental health either for themselves or for people they know have a mental disorder.

3. Phases of the project

This chatbot passes through phases, which are:

1. Searching for an available dataset on web: from well-known web sites like kaggle

The dataset used on this project is available on the following link: Mental Health FAQ for Chatbot .

2. EDA or Exploring Data Analysis: this phase aims to discover and analyse the data within the dataset using pandas functions like describe() to get the necessary statistics about the data and info() to get a description about the dataset itself, like the number of columns it contains and their type.

   In our case, and since there are no numerical data, only text, we did not use describe() function, because it deals only with numbers.

This phase will allow the developers to find any missing values within their dataset and choose the right strategy to deal with those values. In our case, they were no missing values.

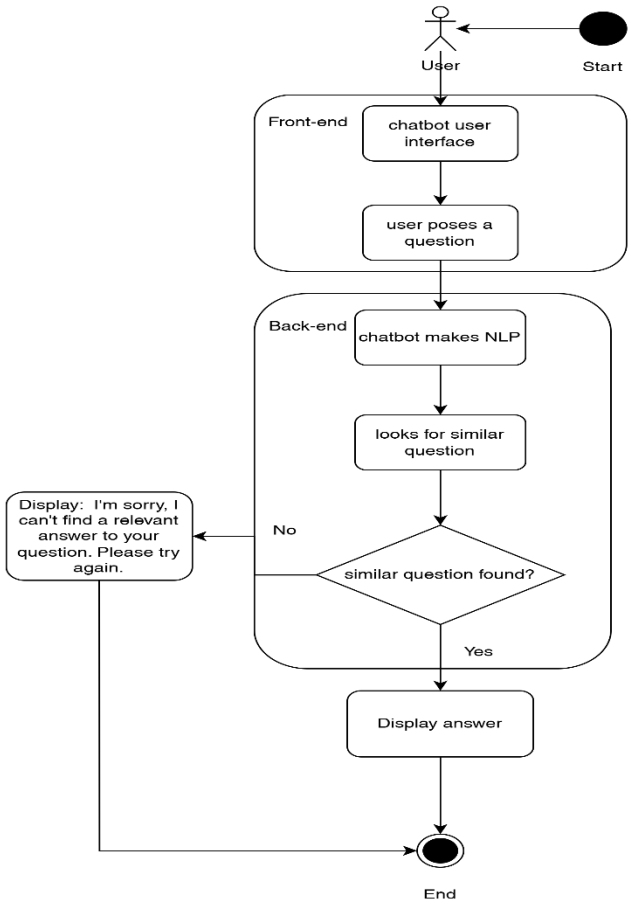
3. NLP or natural processing language: after the EDA phase, it comes the NLP one, which is for role of making the chatbot understands the users expressions by extracting the keywords from their text(in our project the text is in form of question) using the library spacy.

This phase also passes by the following steps:

- Text cleaning: to delete punctuation, signs, numbers and turn the capital letters into small ones in our document.
- Tokenisation: by transforming the obtained cleaned text into tokens (which is an array of words).
- Lemmatisation: by returning the words of the document into their origin.
- Stop-words: to delete unnecessary words and keep the important ones(keywords)
- Tf-Idf-Vectorization: by transforming the keywords into vector of values between ‘’0’’ and ‘’1’’, which is called the score, this step identifies the most important words (which has the highest score in the document) in order to be extracted by the chatbot and used in the next step which is cosine similarity.
- Cosine similarity: after extracting the words with the highest score, the chatbot will compare those words with the ones existing in the dataset, by calculing their cosine, if it is closer to ‘’1’’, that means the question of the user is similar enough with the question in the dataset, and then having the corresponding answer.

4. Creating an interface: to easily interact with the chatbot, using streamlit.
5. Testing the chatbot: by typing the questions and getting their answers.
6. Deploying the chatbot: using pyngrok.

4. Chatbot architecture:



5. Source Code:

```
!pip install streamlit
!pip install pyngrok
!pip install scikit-learn
import re
import spacy
import pandas as pd
nlp = spacy.load("en_core_web_sm")
def text_preprocessing(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = re.sub(r'\s+', ' ', text)
    doc = nlp(text)
    tokens = [token.text for token in doc]
    lemmas = [token.lemma_ for token in doc]
    stopwords_removed = [token.lemma_ for token in doc if not token.is_stop]
    text = " ".join(stopwords_removed)
    return text
df = pd.read_csv('/content/Mental_Health_FAQ.csv')
df.head()
df.info()
df.drop('Question_ID', axis=1, inplace=True)
df.head()
df['Question'] = df['Questions'].apply(text_preprocessing)
df['Answer'] = df['Answers'].apply(text_preprocessing)
print(df[['Answers', 'Answer']].head())
df[['Questions', 'Question']].head()
%%writefile app.py
import streamlit as st
from sentence_transformers import SentenceTransformer
```

```
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd
df = pd.read_csv('/content/Mental_Health_FAQ.csv')
# SBERT
model = SentenceTransformer('all-MiniLM-L6-v2')
corpus_embeddings = model.encode(df["Questions"].tolist())
# Streamlit interface
st.title("Mental Health Chatbot")
user_question = st.text_input("Type your question:")
if user_question:
    user_embedding = model.encode([user_question])
    similarities = cosine_similarity(user_embedding, corpus_embeddings)
    best_match_idx = similarities.argmax()
    best_match_score = similarities[0, best_match_idx]
    question_similaire = df["Questions"].iloc[best_match_idx]
    reponse_similaire = df["Answers"].iloc[best_match_idx]
    if best_match_score > 0.5:
        st.write(f"**Question utilisateur:** {user_question}")
        st.write(f"**Question similaire trouvée:** {question_similaire}")
        st.write(f"**Réponse correspondante:** {reponse_similaire}")
        st.write(f"**Score de similarité:** {best_match_score:.2f}")
    else:
        st.write("I'm sorry, I can't find a relevant answer to your question. Please try again.")
from pyngrok import ngrok
ngrok.set_auth_token("2qcwrweSgTmouvxF5nG76FPQlvO_7wTUMbfHb6Tu62oiNdQKe")
public_url = ngrok.connect(8501)
print("Streamlit est accessible via :", public_url)
!streamlit run app.py &>/content/logs.txt &
```

6. User manual

Our chatbot is accessible through a link, after clicking on it; it will redirect the user to his browser where he can interact with the chatbot by typing a question related to mental health and getting its suitable answer.

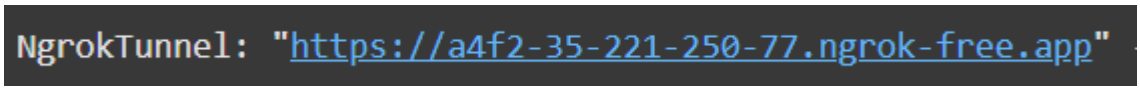


Figure1. Access link

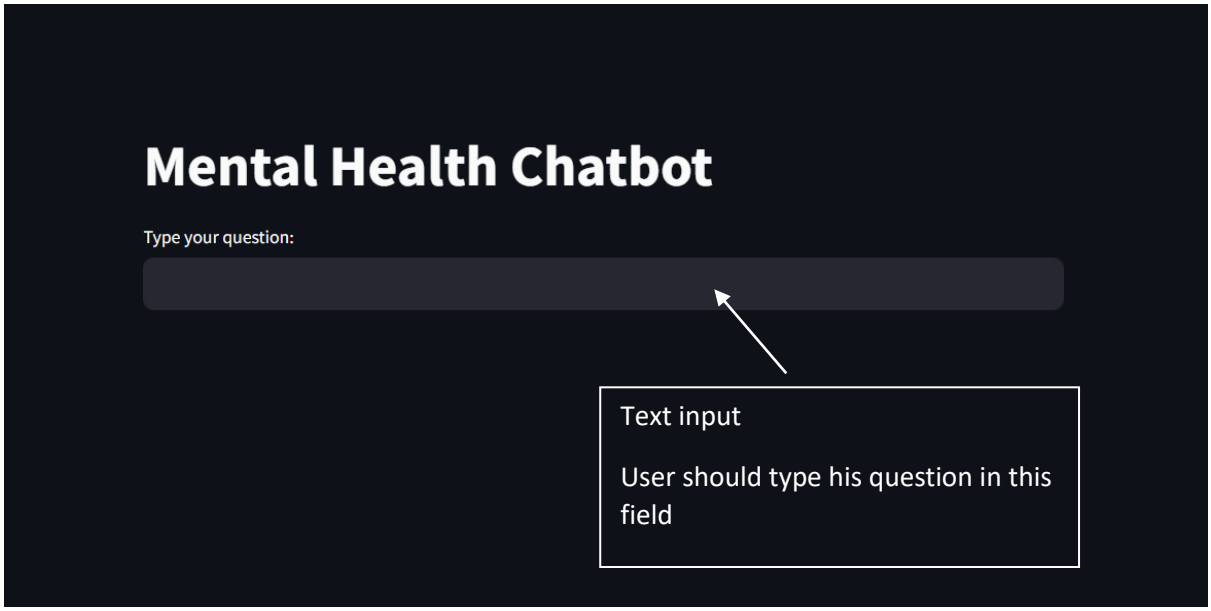


Figure 2. Chatbot interface

Examples:

Figure 3. Example of question about depression

In this figure, we typed a question about depression, and it returned us the suitable answer,

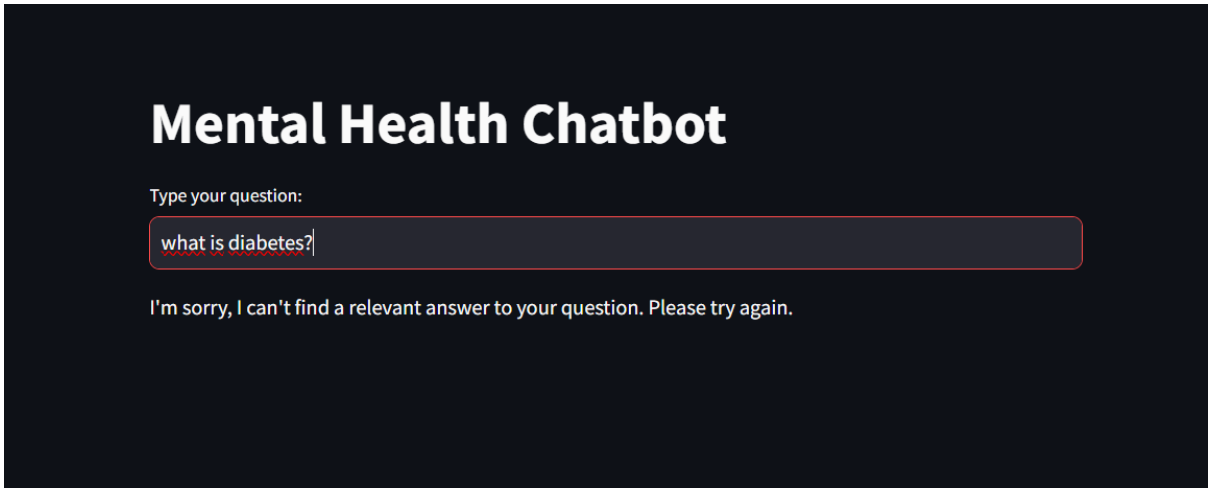


Figure 4. Example of question outside our domain

In this example, we typed a question about diabetes, which is different subject of the mental health domain, and as a result, the chatbot returned the wanted answer.