**Documentation for FAQ Search System using Streamlit and Pyngrok**

This code creates a web-based FAQ (Frequently Asked Questions) search system using Streamlit for the front-end and Pyngrok for tunneling the localhost app to the internet. The search functionality is based on **cosine similarity** using TF-IDF (Term Frequency-Inverse Document Frequency) to match the user’s query with pre-defined FAQ data.

The entire system allows a user to enter a query, searches for the most relevant FAQs from a pre-loaded dataset, and displays them along with a relevance score.

Here is a step-by-step breakdown of the code, what each part does, and how you can run the application:

**Key Components of the Code**

**1. Required Libraries**

* **Streamlit:** A Python framework used to create web applications quickly.
* **Pyngrok:** Used to create a public URL to expose your local application.
* **NumPy:** Used for numerical computations, specifically for sorting and similarity scoring.
* **sklearn (TF-IDF & cosine similarity):** Scikit-learn provides the functionality to convert text to vectors using TF-IDF and calculate the similarity between the user's query and FAQ data.

You will need to install these packages before running the code.

!pip install streamlit

!pip install pyngrok

**2. Ngrok Token Configuration**

Before starting the app, you need to authenticate Ngrok by providing your token. This is required to expose your local Streamlit app to the internet. Replace the placeholder 2nniuSUDub9xNU69jkEqJUb8nJH\_EJXDKVPxVHHLUwRRMPnN with your own Ngrok authtoken.[In this code there is no need to do that,as this authtoken is enough,but in case the free authcode runs out of free service,put your own authtoken]

!ngrok authtoken <your\_ngrok\_authtoken>

**3. Creating and Connecting the Tunnel**

This part of the code uses Pyngrok to expose the Streamlit app running on localhost:8501 to a public URL so that anyone with the link can access the app.

from pyngrok import ngrok

public\_url = ngrok.connect(8501, "http")

print(f"Streamlit is live at: {public\_url}")

**4. FAQ Data**

The FAQ data is hardcoded into the app [In the PS,it has been given that the csv file will be given to us,thus the hardcoding] as a list of dictionaries, where each dictionary contains a **question** and its corresponding **answer**. This data will be used to match and return relevant FAQs based on the user's query.

faq\_data = [

{"question": "What is the process for admission into SARAS AI Institute?", "answer": "The admission process at Saras AI Institute typically involves..."},

...

]

**5. Text Preprocessing with TF-IDF**

* The TfidfVectorizer() converts the FAQ questions into a vector form that reflects the importance of words in each question. This representation is then used to calculate how similar a user's query is to each FAQ question using **cosine similarity**.
* The fit\_transform method is applied to the FAQ questions, converting them into a sparse matrix of TF-IDF values.

vectorizer = TfidfVectorizer()

faq\_vectors = vectorizer.fit\_transform(faq\_questions)

**6. Search Functionality**

The core of the search functionality is the get\_relevant\_faqs() function. Here's how it works:

* **Input:** A user’s query string.
* **Process:** The query is transformed into a vector (using the same TF-IDF model as the FAQs), and then cosine similarity is calculated between the query and all FAQ vectors. This determines how closely the user's query matches each FAQ.
* **Output:** It returns the top 3 FAQs that are most relevant to the query, sorted by their similarity score.

def get\_relevant\_faqs(query, top\_n=3):

query\_vector = vectorizer.transform([query])

similarities = cosine\_similarity(query\_vector, faq\_vectors).flatten()

top\_indices = np.argsort(similarities)[-top\_n:][::-1]

relevant\_faqs = []

for idx in top\_indices:

relevant\_faqs.append({

'question': faq\_questions[idx],

'answer': faq\_data[idx]['answer'],

'score': similarities[idx]

})

return relevant\_faqs

**7. Streamlit Web App**

* **st.title**: This creates a title for the Streamlit app.
* **st.text\_input**: The user inputs their query in a text box.
* **st.write**: Once a query is entered, the relevant FAQs (and their scores) are displayed on the web interface.

st.title("FAQ Search System")

# Input from the user

user\_query = st.text\_input("Enter a query:")

if user\_query:

top\_relevant\_faqs = get\_relevant\_faqs(user\_query)

st.write("### Top Relevant FAQs:")

for i, faq in enumerate(top\_relevant\_faqs, 1):

st.write(f"\*\*{i}. Question:\*\* {faq['question']}")

st.write(f"\*\*Answer:\*\* {faq['answer']}")

st.write(f"\*\*Relevance Score:\*\* {faq['score']:.2f}\n")

**How the Code was modelled to run as a website**

**Step 1: Install Required Libraries**

Run the following commands to install the necessary packages for the app:

!pip install streamlit

!pip install pyngrok

**Step 2: Ngrok Authentication**

You need an Ngrok account to get an authentication token. After registering, get your authtoken from the Ngrok dashboard and run the following command:

!ngrok authtoken <your\_ngrok\_authtoken>

**Step 3: Running the Streamlit App**

Once the code is written and libraries installed, you need to run the Streamlit app.

1. Write the app code into a Python file named app.py using the following command:

%%writefile app.py

Then paste the entire Streamlit app code inside the file.

1. Start the Streamlit app:

!streamlit run app.py &>/dev/null&

**Step 4: Create a Public URL**

Finally, expose the localhost app via Ngrok by running this command:

public\_url = ngrok.connect(8501, "http")

print(f"Streamlit is live at: {public\_url}")

This will provide you with a publicly accessible URL where you or anyone else can access your FAQ search system.

NOTE[For the evaluator]-Just run the entire code at once and click on the url printed at the very end like “Streamlit is live at:{url}”. Then ask any question related to saras ai institute to test the validity of the solution code.

**How the System Works**

1. The user enters a query on the web page.
2. The app processes this query by converting it into a TF-IDF vector.
3. It computes the cosine similarity between the query vector and pre-loaded FAQ questions.
4. The top 3 most similar FAQ questions and their corresponding answers are displayed along with a relevance score.