PRESENTATION

Mental Health FAQ Assistant

A Comprehensive AI-powered FAQ System using Python, Sentence Transformers, FAISS, and Gradio.

Created by: TokenX

AI-Powered Mental Health Support

Table of Contents

Project Overview

Goal

Create a robust Mental Health FAQ assistant to answer user questions efficiently, providing timely access to mental health information.

Approach

Utilizes modern Natural Language Processing and vector search technology for rapid and relevant answer retrieval based on semantic similarity.

Technology Stack

Python, pandas, sentence-transformers, FAISS, Gradio

System Architecture

- **CSV FAQ Dataset** as input
- **Data preprocessing** with pandas
- **Question embedding** using Sentence Transformers
- **Similarity search** using FAISS index
- **User interaction** via Gradio UI

CSV Data

Preprocessing

Embeddings

FAISS Index

Gradio UI

Data Preparation

1 CSV FAQ Upload

Utilize Google Colab's files.upload() function to get the Mental Health FAQ dataset into the environment.

```
from google.colab import files
uploaded = files.upload()
```

2 Load with Pandas

Import the dataset using pandas for efficient data manipulation and analysis.

3 Preview Data Structure

Inspect column names and first few rows to understand the dataset's organization.

4 Standardize Column Names

Rename columns to follow a consistent naming convention for downstream processing.

Loading and Exploring Data

Code Example

```
# Import pandas library
import pandas as pd

# Load the CSV file
data = pd.read_csv('Mental_Health_FAQ.csv')

# Display column names
print(data.columns.tolist())

# Preview first few rows of data
data.head()
```

Purpose

Ensure data integrity and inspect column structure before proceeding with model development. This step allows us to understand:

Standardizing and Verifying Columns

Column Renaming

'Questions' → 'question'

'Answers' → 'answer'

Purpose

Standardization ensures consistent naming conventions for easy access and automation in downstream processing.

Code Example:

```
data.rename(columns={'Questions': 'question', 'Answers': 'answer'}, inplace=True)
print(data.columns.tolist()) # Should be ['Question_ID', 'question', 'answer']
```

Consistent naming patterns simplify code readability and make future development more intuitive.

Building Embeddings with Sentence Transformers

Sentence Transformer Model

Using the 'all-MiniLM-L6-v2' model which creates 384-dimensional embeddings that capture semantic meaning from text.

Vector Representation

Converts text questions into numeric vectors that capture semantic meaning, enabling similarity-based search.

Code Implementation:

```
from sentence_transformers import SentenceTransformer

# Load embedding model
model = SentenceTransformer('all-MiniLM-L6-v2')

# Generate question embeddings
faq_embeddings = model.encode(data['question'].tolist())
```

Creating the FAISS Index

FAISS (Facebook AI Similarity Search) enables efficient and scalable similarity search and clustering of dense vectors.

Perfect for our use case: quickly retrieving FAQ questions similar to user queries based on embedding vector proximity.

Python

```python
import faiss
import numpy as np

# Create FAISS index
faq_index = faiss.IndexFlatL2(faq_embeddings.shape[1])

# Add FAQ question embeddings to index
faq_index.add(faq_embeddings)

# Verify index size
print("✅ FAISS index built with", len(data), "questions.")
```

Query Function Implementation

Query Embedding Generation

User queries are converted to vector embeddings using the same sentence transformer model.

FAISS Similarity Search

The FAISS index efficiently retrieves top-k similar questions from the dataset based on vector distance.

Results Display

The function returns the most relevant question-answer pairs for the user query.

Function Implementation:

```
def answer_query(query, k=3):
    query_embedding = model.encode([query])
    D, I = faq_index.search(np.array(query_embedding), k)

    print(f"\n📌 Your Query: {query}")
    for rank, idx in enumerate(I[0]):
        question = data.iloc[idx]['question']
        answer = data.iloc[idx]['answer']
        print(f"\n🔹 Match #{rank+1}:\nQ: {question}\nA: {answer}")
```

Sample Queries and Results

Query 1: "What are the symptoms of mental illness?"

Match #1:

Q: What are common signs of mental health issues?
A: Common signs include persistent sadness, excessive fears, withdrawal from social activities, significant mood changes, sleep disturbances, and difficulty concentrating...

Match #2:

Q: When should I seek professional help for mental health concerns?
A: Consider seeking professional help when symptoms persist for weeks, interfere with daily activities, cause significant distress, or involve thoughts of self-harm...

Query 2: "How do people recover from depression?"

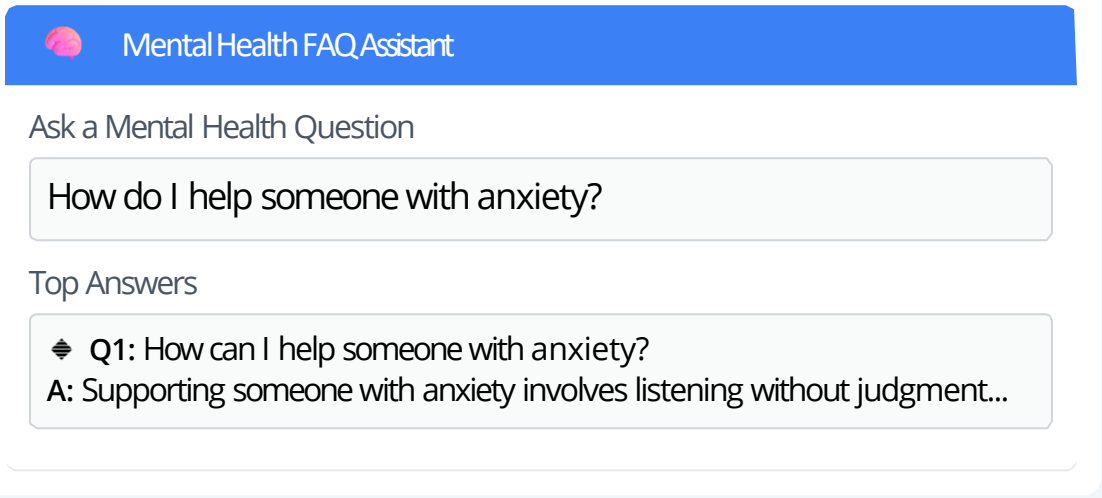
Match #1:

Q: What treatments are effective for depression?

Gradio UI Integration & Key Features

Gradio Interface

```
gr.Interface(
    fn=gradio_faq,
    inputs=gr.Textbox(label="Ask a Mental Health Question"),
    outputs=gr.Markdown(label="Top Answers"),
    title="🧠 Mental Health FAQ Assistant",
    description="Type your question to get relevant answers."
).launch()
```



Key Features

Fast Semantic Search
FAISS enables near-instantaneous retrieval from large FAQ databases using vector similarity.

Easily Extensible
Simply update the CSV with new questions and answers, then rebuild the index to expand capabilities.

Healthcare-Themed UI
Professional design instills confidence and comfort for users seeking mental health information.

Intuitive User Experience
Simple question input with clearly formatted answers makes information easily accessible.

Future Work & Enhancements

Expand FAQ Dataset

Cover additional mental health topics and incorporate expert-verified responses for comprehensive support.

Improve Answer Ranking

Enhance similarity algorithms and implement answer summarization for more precise and concise responses.

Multi-language Support

Implement cross-lingual embeddings to support queries and responses in multiple languages.

Web Service Deployment

Deploy as a web service, mobile application, or integrate into existing mental health platforms and chatbots.

Continuous Improvement

Monitor user queries and feedback to identify gaps in coverage and improve system performance over time.