Name: Sachin Sharma

Reg.no- 12115786

Project-link: https://huggingface.co/spaces/sachin98/smart-space

Introduction

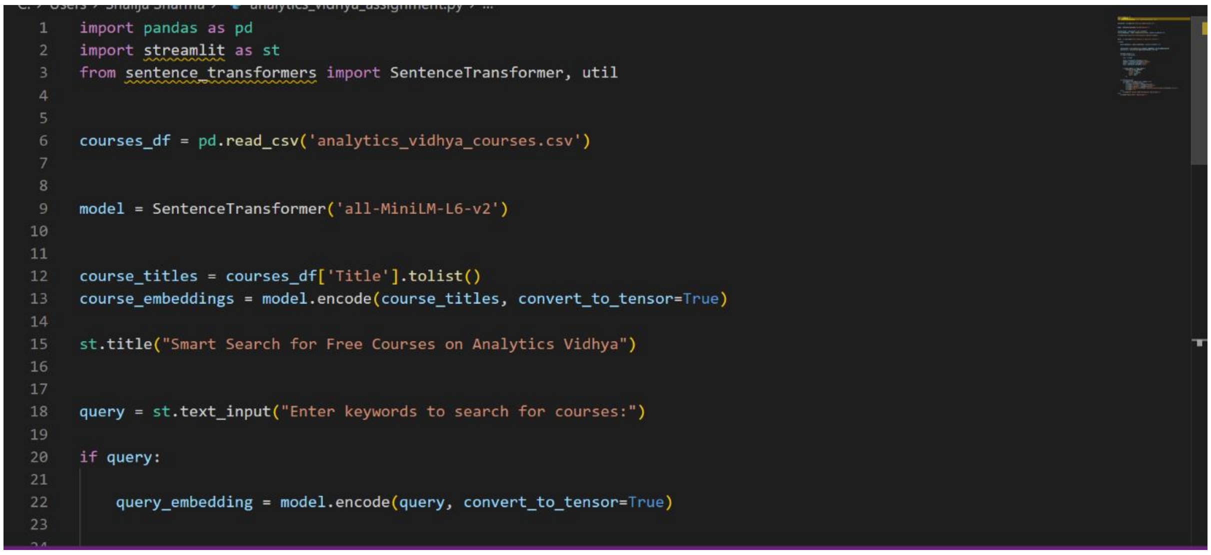
The objective of this project is to develop a smart search tool that assists users in finding relevant free courses on the Analytics Vidhya platform. With an increasing number of online courses, a robust search functionality is crucial to enhance user experience and accessibility.

Data Collection

The data for this project was collected from the Analytics Vidhya free courses page using web scraping techniques. The primary data fields extracted include:

- **Title**: The name of the course.
- **Description**: A brief overview of the course content.
- **Lessons**: The number of lessons in the course.
- **Price**: The cost of the course (in this case, all courses are free).
- **Link**: The URL for enrollment.

The data extraction was performed using the BeautifulSoup library in Python. Below is a snippet of the code used for this task:



```python
import pandas as pd
import streamlit as st
from sentence_transformers import SentenceTransformer, util


courses_df = pd.read_csv('analytics_vidhya_courses.csv')


model = SentenceTransformer('all-MiniLM-L6-v2')


course_titles = courses_df['Title'].tolist()
course_embeddings = model.encode(course_titles, convert_to_tensor=True)

st.title("Smart Search for Free Courses on Analytics Vidhya")


query = st.text_input("Enter keywords to search for courses:")

if query:

    query_embedding = model.encode(query, convert_to_tensor=True)

```

Data Processing

After collecting the data, the next step involved cleaning and processing it to ensure consistency and usability. This included:

- **Handling Missing Values**: Any courses without descriptions were assigned a placeholder.
- **Data Formatting**: Ensured that all text fields were stripped of unnecessary whitespace and standardized in format.

- **Encoding**: The course titles were prepared for generating embeddings.

1 Embedding Model: For this project, I selected the SentenceTransformer model, specifically the all-MiniLM-L6-v2 variant. This choice was based on the following criteria:

- **Efficiency**: The model is lightweight and provides high-quality embeddings, making it suitable for real-time applications.
- **Semantic Understanding**: The embeddings capture the semantic meaning of the course titles, enabling effective query matching.

2 LLM Selection

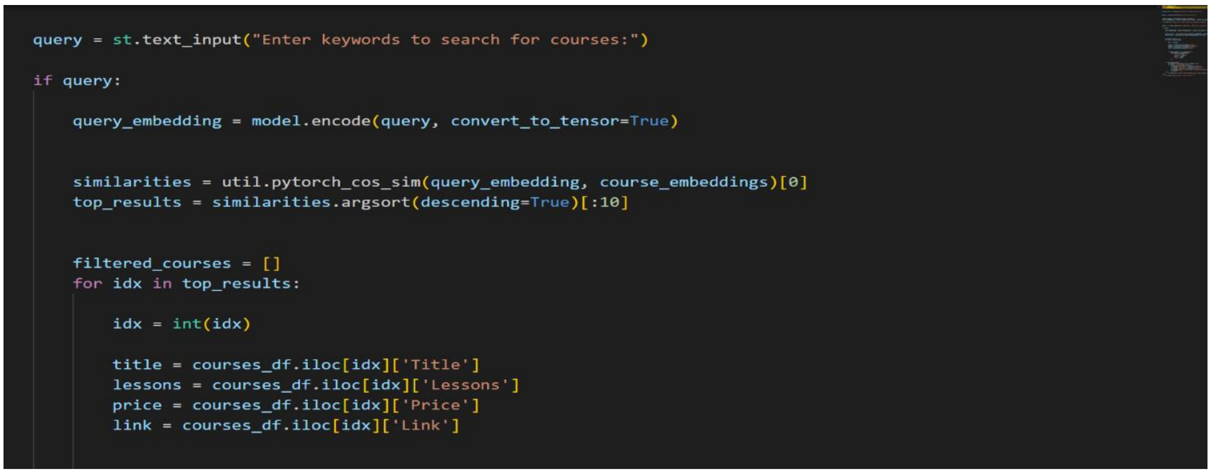
While this project primarily focuses on using embeddings for search functionality, if an LLM were to be integrated in the future, models like GPT-3 could be considered for enhancing user queries and providing richer context-aware responses.

Search Methodology

The search functionality utilizes cosine similarity to match user queries with course titles. When a user enters a keyword:

1. An embedding is generated for the user query.
2. Cosine similarity scores are calculated between the query embedding and all course title embeddings.
3. The top results are retrieved based on the highest similarity scores.

The following code snippet illustrates this methodology:



```
query = st.text_input("Enter keywords to search for courses:")

if query:

    query_embedding = model.encode(query, convert_to_tensor=True)


    similarities = util.pytorch_cos_sim(query_embedding, course_embeddings)[0]
    top_results = similarities.argsort(descending=True)[:10]


    filtered_courses = []
    for idx in top_results:

        idx = int(idx)

        title = courses_df.iloc[idx]['Title']
        lessons = courses_df.iloc[idx]['Lessons']
        price = courses_df.iloc[idx]['Price']
        link = courses_df.iloc[idx]['Link']
```

Conclusion

The smart search tool developed for free courses on Analytics Vidhya effectively utilizes embeddings and cosine similarity to enhance the search experience. The project not only demonstrates the power of semantic search but also lays the groundwork for future enhancements, such as integrating LLMs for more advanced interactions.