Smart Search Tool for Analytics Vidhya Free Courses

Table of Contents

1. Introduction

The goal of this project is to build a smart search tool that enables users to efficiently find the most relevant free courses on the Analytics Vidhya platform. This tool leverages natural language processing (NLP) techniques to improve search relevance and user experience.

2. Data Collection

Course Data Gathering

Utilize web scraping or direct API calls (if available) to gather information on free courses. Key data points include:

- **Course Title**
- **Course Description**
- **Curriculum** (topics covered in the course)
- **Instructor(s)**
- **Duration**

Example of Course Data Structure:

```
json

[
    {
```

```
        "title": "Introduction to Data Science",
        "description": "Learn the basics of data science.",
        "curriculum": ["Data Analysis", "Statistics", "Machine Learning"],
        "instructor": "John Doe",
        "duration": "4 weeks"
    },
    ...
]
```

Tools Used

- **BeautifulSoup** (for web scraping)
- **Pandas** (for data manipulation)

3. Data Preprocessing

Clean and prepare the collected data for embedding using the following steps:

- Remove any duplicates.
- Normalize text (lowercase, remove special characters).
- Split curricula into separate entries for better indexing.

Example of Preprocessing Code

```
python

import pandas as pd

# Load the dataset
data = pd.read_json('courses.json')

# Normalize text
data['title'] = data['title'].str.lower()
data['description'] = data['description'].str.lower()
```

4. Embedding Model Selection

Choose an appropriate embedding model to convert course data into vectors. Consider using:

- **OpenAI's Text Embeddings**: Effective for natural language understanding.
- **Sentence Transformers**: For context-aware embeddings.

Model Selection Reasoning

Both models are capable of capturing semantic similarity between user queries and course descriptions, which is crucial for relevant search results.

5. Search System Implementation

System Architecture

- **Embedding Generation**: Convert course data into vector embeddings using selected models.
- **Vector Database**: Store embeddings in a vector database like **Faiss** or **Pinecone** for efficient similarity searches.
- **Search Algorithm**: Implement a search algorithm that retrieves similar course embeddings based on user queries.

Code Snippet for Embedding Generation

```
python

from sentence_transformers import SentenceTransformer

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

# Generate embeddings
data['embedding'] = data['description'].apply(lambda x: model.encode(x))
```

Search Implementation Code

```
python

import faiss

# Create a FAISS index
index = faiss.IndexFlatL2(embeddings_dimension)
index.add(embeddings_array)

def search_courses(query):
    query_embedding = model.encode(query)
    D, I = index.search(query_embedding, k=5)  # Retrieve the top 5 results
    return data.iloc[I[0]]
```

6. Deployment on Huggingface Spaces

Framework Selection

Use **Gradio** or **Streamlit** to build the user interface for the smart search tool.

Deployment Steps

1. Create a Huggingface account and log in.
2. Set up a new space and select **Gradio** as the framework.

3. Upload the search tool code, including necessary dependencies (e.g., `sentence-transformers`, `faiss`, `gradio`).
4. Ensure the user interface allows input for queries and displays results seamlessly.

Example Gradio Interface Code

```
python

import gradio as gr

def search_interface(query):
    results = search_courses(query)
    return results[['title', 'description']].to_dict(orient='records')

gr.Interface(fn=search_interface,
             inputs="text",
             outputs="json").launch()
```

7. User Guide

How to Use the Smart Search Tool

1. Navigate to the deployed Huggingface Spaces link.
2. Enter a keyword or phrase related to the desired course in the search input.
3. Click "Search" to view relevant courses listed as results.

8. Conclusion

This documentation outlines the step-by-step approach taken to create a smart search tool for finding relevant free courses on the Analytics Vidhya platform. The implementation of a vector-based search system significantly enhances the user experience by delivering accurate results based on natural language queries.

9. References

- Analytics Vidhya Free Courses
- Sentence Transformers Documentation
- Faiss Documentation