Name: **DEVIKA PEMMADI**

Reg. No: **Y24MC20046**

# University: ACHARYA NAGARJUNA UNIVERSITY

Course : **MASTER OF COMPUTER APPLICATIONS(MCA)**

GitHub Work Link:

[**https://github.com/Devika-15/Smart-SearchFeature-Analytics-Vidhya**](https://github.com/Devika-15/Smart-SearchFeature-Analytics-Vidhya)

**Task**

**Here’s what you need to do:**

1. Browse through the free courses on the platform: Explore and collect data (e.g., course titles, descriptions, curriculum) from the free courses available on Analytics Vidhya platform.
2. Develop a smart search system: Build a search tool that quickly finds and suggests the most relevant courses using keywords and natural language queries. You should use Generative AI methods/tools like vector embeddings for querying and an LLM to get the results.
3. Deploy the tool on Huggingface Spaces: Once your search tool is ready, deploy it on Huggingface Spaces. You can use any framework (such as Gradio, Streamlit, or others) as long as it is publicly accessible via Huggingface Spaces.
4. Share the link for review: Provide us with the public link to your deployed smart search tool for evaluation.
5. Share your approach: Share with us a document (e.g. docx, pdf, etc.) that explains how you have approached the problem, and the methodology behind the embedding model and LLM selection.
6. The more advanced and effective your search tool is, the better your chances of standing out.

**Approach:**

**1. Data Collection:**

* **Course Titles**:
  + **Reason**: The course title serves as the primary identifier for each course. It gives users a clear and concise understanding of what the course is about. Including this information is crucial for any search or filtering functionalities, as users often search for specific topics or titles.
* **Descriptions**:
  + **Reason**: Descriptions provide an overview of what the course covers, its objectives, and any prerequisites. They help users quickly assess whether the course aligns with their interests and learning goals. Detailed descriptions enhance the search experience by offering context around the course content.
* **Links**:
  + **Reason**: Links provide direct access to the course details on the Analytics Vidhya website. Including this information is crucial for the search tool, as it allows users to quickly navigate to the course page for more in-depth information, such as enrollment options, curriculum, and instructor details. This enhances the usability of the smart search tool by facilitating easy access to relevant resources.

# 2. Development of Smart Search System:

After successfully collecting the course data, the next step was to Build a tool that efficiently retrieves and suggests relevant courses based on user queries.

1. **Embedding-based Search:**

* Using vector embeddings to represent course data and user queries
* The **SentenceTransformer** model (all-MiniLM-L6-v2) was initialized again, as it is necessary for generating embeddings from user queries.

1. **Natural Language Processing (NLP):**

* Incorporate a Large Language Model (LLM) such as GPT or similar to process user queries.
* Leverage the LLM for query expansion, semantic understanding, and ranking.

1. **System Design:**

* Input: User’s keyword or natural language query.
* Processing: Convert query and course descriptions into vector embeddings and find the nearest matches.
* Output: Return the top N most relevant courses based on cosine similarity or other distance metrics.

1. **Example Query**:

* An example search query was provided, such as "data science beginner course," to demonstrate how the function can be utilized. The results are printed, displaying the most relevant courses based on the user's query.

**3. Deployment on Huggingface Spaces**:

In this step, we Make the tool publicly accessible by deploying it on Huggingface Spaces.

1. **Importing Necessary Libraries**:

* The gradio library is imported to create the web interface.
* The custom  top\_results function from the SmartSearchEngine module is imported to facilitate searching.

1. **Defining the Search Function**:

* A function named smart\_search is defined to serve as a bridge between the Gradio interface and the underlying search logic. This function takes a user query as input, calls the top\_results function with the query, and returns the results.

1. **Creating the Gradio Interface**:

* An instance of gr.Interface is created to define the user interface:
* **fn**: Specifies the function to be called when the user submits a query (smart\_search).
* **inputs**: Defines the input component, which is a text box allowing users to enter their search query. It has two lines and a placeholder text to guide users.
* **outputs**: Sets the output format to display the search results , making it easy for users to read the information.
* **title** and **description**: Provide a title and description for the interface to enhance user understanding.

1. **Launching the Interface**:

* The iface.launch(share=True) method is called to start the Gradio interface. The share=True argument allows the interface to be accessible via a public link, enabling others to use the tool without needing to set up any local environment.

**4. Screenshot of the Deployed application:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated