QueryTube:AI_SemanticSearchTube: Building a Semantic Search App with YouTube Data

Project statement:

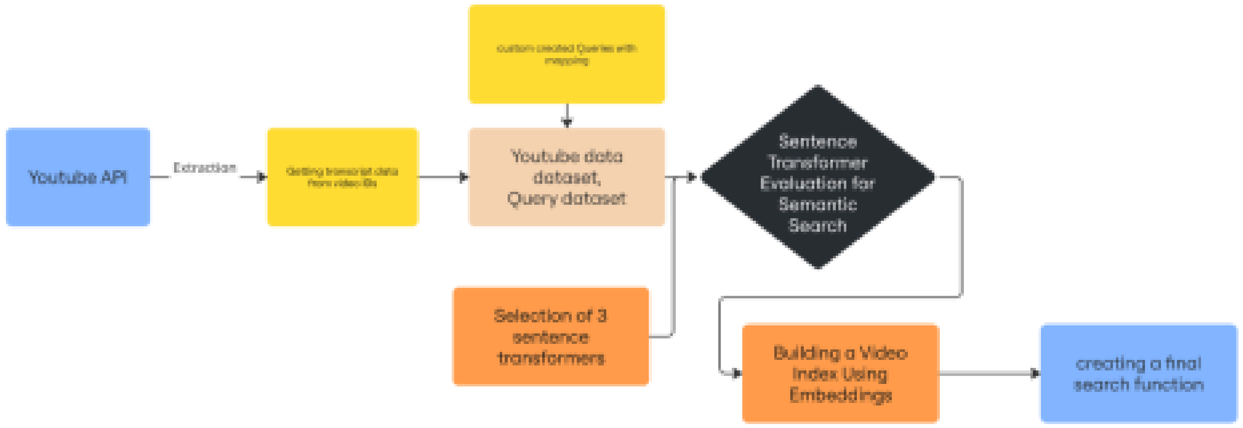
The goal of this project is to build a YouTube semantic search system by extracting and transforming video data (title, date, transcript) via YouTube APIs. The system will allow users to input natural language queries and receive the top-5 most semantically relevant video titles or IDs. Interns will gain hands-on experience in NLP, data engineering, and information retrieval by building a complete semantic search engine.

Outcomes:

● Extract and preprocess metadata and transcripts from YouTube videos using APIs. ● Understand and apply transformer-based text embedding models. ● Evaluate similarity and distance metrics for effective semantic retrieval. ● Build and deploy a complete semantic video search pipeline.

Milestones:

Milestone 1: YouTube Data Collection and API Mastery (Weeks 1–2)
Milestone 2: Transcript Extraction and Data Cleaning (Weeks 3–4)
Milestone 3: Sentence Transformer Evaluation for Semantic Search (Week 5-6)
Milestone 4: Implementing and Tuning Semantic Search function (Week 7-8)


custom created Queries with mapping
Youtube API
Extraction
Getting transcript data from video IDs
Youtube data dataset, Query dataset
Sentence Transformer Evaluation for Semantic Search
Selection of 3 sentence transformers
Building a Video Index Using Embeddings
creating a final search function

Milestone 1: YouTube Data Collection and API Mastery (Weeks 1–2)

Module-1: YouTube API Fundamentals and Video Metadata Extraction

Tasks:

- Set up a Google Cloud project and generate an API key
- Choose a YouTube channel (with 100–350 diverse videos; avoid repetitive content like stock updates)
- Learn how to use the endpoint: https://www.googleapis.com/youtube/v3/search

Key Parameters:

- o channelId: ID of the selected channel
- o maxResults: Up to 50 per request
- o part: ["snippet", "id"]
- o order: “date”
- o pageToken: for pagination

Deliverables:

- Python script to extract all videos from a selected channel using pagination logic
- Custom function to extract video ID, published date, and title.

Learning Outcomes:

- Understand the YouTube Data API v3
- Authenticate and interact with the API using an API key
- Fetch video metadata programmatically

Module 2: Exploratory Data Analysis (EDA) and Structuring Video Metadata

Tasks:

- Store extracted data in Polars or Pandas dataframe
- Analyze:
  - o Row and column uniqueness
  - o Title distributions
  - o Publish frequency over time
- Perform EDA on metadata (date distribution, title uniqueness, missing values)

Deliverables:

- Notebook or script with exploratory charts/tables
- Cleaned video metadata dataset

Milestone 2: Transcript Extraction and Data Cleaning (Weeks 3–4)

Module 3: Extracting Video Transcripts with YouTube Transcript API

Tasks:

- Use the youtube_transcript_api Python package to fetch auto-generated captions or transcripts
- Use appropriate methods to collect and organize transcript text
- Identify and log videos without available transcripts

Deliverables:

- Dataframe enriched with a transcript column
- Transcript extraction logic and logs of failed extractions

Module 4: Cleaning and Normalizing Transcripts

Tasks:

- Standardize titles and transcripts by removing or replacing special characters
- Handle null or missing transcript entries
- Generate an initial list of ~70–80 search queries (topics, keywords, or phrases relevant to the selected videos) and map them to
- Familiarize with at least three SentenceTransformer models (e.g., all-MiniLM-L6-v2) from hugging face or https://sbert.net/
- Understand sentence-transformers and their role in semantic search. ●
Prepare evaluation setup to compare semantic similarity performance

Deliverables:

- Cleaned dataset with video_id, title, datetime, transcript
- Preliminary set of evaluation queries for Week 5
- Summary of semantic search understanding and model selection rationale

Milestone 3: Sentence Transformer Evaluation for Semantic Search(Week 5-6)

Module 5: Cleaning and Normalizing Transcripts

Tasks:

- Use SentenceTransformer to embed video titles and transcripts using three candidate models
- Use the pre-defined 70–80 search queries and embed them

- Compare distance-based (euclidean, manhattan, chebyshev) and similarity-based (cosine similarity, dot product) ranking methods
- Evaluate similarity between queries and video transcripts/titles and check how well each model ranks the correct video for each query

Deliverables:

- Evaluation summary: model performance across metrics (e.g., average rank, top-1, top-3 recall)
- Identify the best-performing model and method for semantic retrieval
- Visual or tabular comparison of results across models

Module 6: Building a Video Index Using Embeddings

Tasks:

- Choose the best SentenceTransformer model from Week 5
- Embed the titles and transcripts of each video
- Concatenate and append these embeddings to the original dataset
- Save the final dataframe using Polars or pandas

Deliverables:

- Polars or pandas dataframe with ~768+ embedding features per video
- Parquet or csv file storing complete video index for search
- Sample visualization of embedding structure or dimensionality reduction (optional)

Milestone 4: Implementing and Tuning Semantic Search and creating a final search function (Week 7-8)

Module 7: Optimizing result quality

Tasks:

- Load the video index and selected transformer model
- Encode incoming user query
- Use distance metrics (e.g., manhattan, euclidean, cosine) from sklearn to compute similarity between query embedding and stored vectors ● Rank the closest matches using title and transcript embeddings jointly ● Understand thresholds, top_k rankings, and filtering logic
- Tune the threshold and top_k to optimize result quality

Deliverables:

- Working search function returnSearchResults(query, df)
- Evaluation of different thresholds and distance types
- Sample query-to-result demo in notebook format

Module 8: Final Deployment & Search Interface

Tasks:

- Use the optimized model + distance metric pair
- Create a Python script or notebook that loads the index, runs query search, and returns top-5 matches
- Build an interactive Gradio interface with embedded video previews and markdown descriptions.
- Prepare summary presentation and code documentation.

Deliverables:

- Final query-to-top-5 search engine function
- Gradio interface that takes a query and displays top-5 embedded YouTube videos
- Code submission on GitHub.