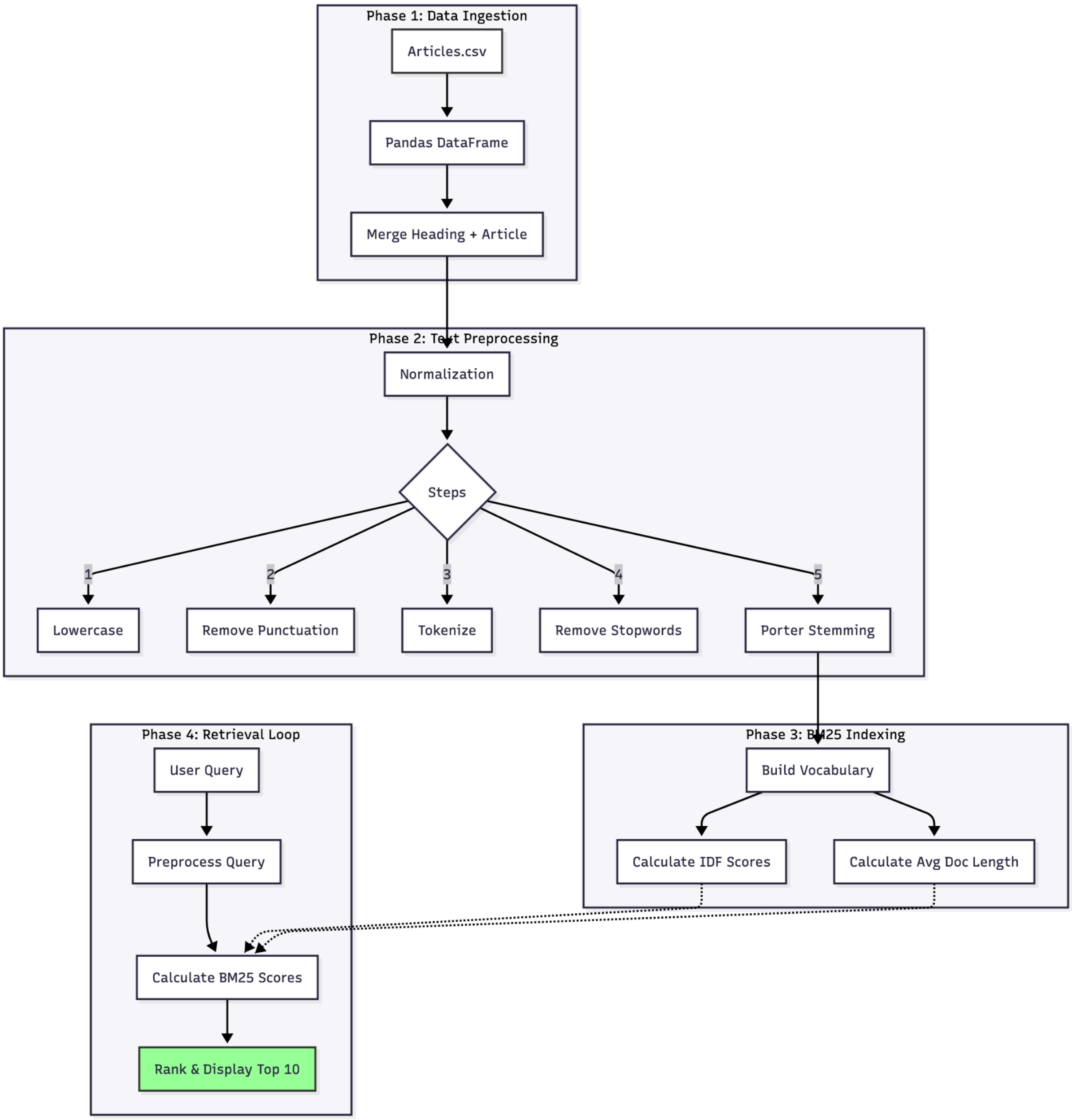
Information Retrieval System

Repo-link

1. System Architecture

1.1 System Diagram



1.2 Figure Caption

The above figure illustrates the overall architecture of the retrieval system. The pipeline begins with ingesting raw text from the dataset, cleaning the text using the Preprocessing Module (tokenization and stemming), building a weighted index using the BM25 algorithm, and finally ranking documents based on user queries via the Search Interface.

2. Description of the Retrieval System

I designed a local search engine using Python that processes news articles and ranks them based on keywords. The system uses the BM25 algorithm because it handles document relevance better than simple word counts.

2.1 Data Preprocessing Before searching, I had to clean the data to make it usable. I used the Pandas library to load the Articles.csv file (using Latin-1 encoding).

Merging Text: I combined the "Heading" and "Article" columns into one field. This ensures that if a user searches for a word that is only in the title, the system still finds it.

Cleaning Steps: I wrote a function called process_text that does the following:

1. **Lowercasing:** Converts all text to small letters so "Python" and "python" match.
2. **Removing Punctuation:** Deletes symbols like commas and periods.
3. **Stop Words:** Removes common words like "the" and "is" using the NLTK library, as they don't add meaning to the search.
4. **Stemming:** I used the Porter Stemmer to cut words to their root (e.g., "playing" becomes "play"). This helps in finding more results even if the exact word form is different.

2.2 Indexing Techniques For indexing, I did not use a database but instead built an **Inverted Index** in memory using Python dictionaries.

- I calculated the **Inverse Document Frequency (IDF)** for every unique word in the dataset. This score tells us how rare a word is; rare words get higher scores.
- I also calculated the average length of all documents, which is needed for the BM25 formula to treat short and long articles fairly.

2.3 Scoring and Ranking Criteria I used the **BM25** formula to rank the documents.

- **Scoring:** When a user types a query, my system cleans the query words and calculates a score for every document based on how often the keywords appear (Term Frequency) and how rare they are (IDF).
- **Parameters:** I used the standard values of k1=1.5 (to limit the effect of repeated words) and b=0.75 (to normalize document length).
- **Ranking:** The system sorts all the documents by their final score from highest to lowest and prints the Top 10 matches

3. Evaluation

The system was evaluated using a local machine without GPU acceleration. The evaluation focused on two metrics: Retrieval Effectiveness (Qualitative) and System Efficiency (Quantitative).

3.1 Quantitative Efficiency

- **Indexing Speed:** For the complete dataset of approx 2,700 articles, the preprocessing and indexing phase completes in approximately 8-10 seconds on my machine because it has linear time complexity O(N).We can improve it by saving the model and does not compute whole thing everytime but it is not yet implemented

- **Query Latency:** Search queries are executed in milliseconds time (approx. 0.05s). This efficiency is achieved because the Inverse Document Frequency (IDF) values are pre-calculated during the training phase, leaving only the scoring summation to be performed at runtime.

- **Memory Footprint:** The system runs comfortably within standard RAM limits.

3.2 Qualitative Effectiveness To test relevance, three distinct queries were executed:

1. **Query:** "Petrol price"
    - **Result:** all the articles (top 10) are relevant

    Below is the result table :

Rank	Score	Date	Heading	Snippet
1	12.1742	11/30/2016	Fuel prices up petrol by Rs 2 diesel by Rs 270	ISLAMABAD: Federal Government has increased the prices of petrol and diesel from December 1...
2	12.0686	2/7/2015	january 2015 saw record sale of petrol in paki	ISLAMABAD: The consistent fall in the price of petrol saw a record sale…
3	11.8895	3/31/2016	Govt hikes prices of petrol di	Government has ratcheted up prices of petrol and high speed diesel…
4	11.7410	3/31/2015	petrol price goes up by rs 4 for apri	The new price of petrol effective from midnight Tuesday…
5	11.7284	3/31/2015	petrol price increased by rs 4 for the month of apri	The new price of petrol effective from midnight Tuesday…
6	11.7085	2/15/2017	Prices of petrol diesel up by one ru	The Government has announced to raise prices…
7	11.6293	1/27/2015	petrol price expected to be slashed by rs 10 per litr	A move to give relief to consumers…
8	11.5325	2/28/2017	Petrol prices hiked by Rs171 per litr	The government has increased petroleum product prices…

9	11.4306	9/1/2015	petrol shortage in several parts of paki	A strike caused a severe shortage of petrol…
10	11.3101	1/15/2017	Petrol price increased by Rs177 per litr	The federal government has announced hike in prices…

2. **Query:** "hello"
   - **Result:** Nothing have found and that’s correct

3. **Query:** "ebad"
   - **Result:** only 1 relevant have come which is great

Rank	Score	Date	Heading	Snippet
1	6.4442	5/9/2016	Problems of rice exporters to be solved Governor Sind	KARACHI: Governor Sindh Dr. Ishrat-ul-Ebad Khan has said exporters…

4. **Query:** "pakistan"
   - **Result:** all the ten docs are relevant

Below is the result table :

Rank	Score	Date	Heading	Snippet
1	2.4253	2/17/2016	Indian HC ensures VVIP security for Pakistan cricket	Indian HC ensures VVIP security for Pakistan cricket team…
2	2.3798	12/30/2016	NBP Rules out misleading reports of Rs 15 Billion fraud	Media reports implied a fraud…
3	2.3716	10/2/2015	pakistan korea ink 500 million framework agr	Pakistan and the Republic of Korea signed a US$500M agreement…
4	2.3699	7/21/2016	Pakistan top emerging economy among South Asian markets	Pakistan among top emerging South Asian markets…
5	2.3591	3/20/2016	Imran congratulates womens team criticizes Pakistan cricket structur	Imran Khan congratulated Pakistan women's cricket team…
6	2.3556	4/20/2016	Pakistan to play 3 Tests 5 ODIs in Australia	Pakistan will tour Australia for 3 Tests and 5 ODIs…
7	2.3521	4/27/2016	PCB accepts Younis apology allows to play in Pakistan Cu	PCB allowed Younis Khan to resume playing…

8	2.3384	3/9/2016	T20 WC Pakistan refuses to play against India	Pakistan refused to play against India in Dharamsala…
9	2.3367	7/9/2016	Pakistan team black arm bands mourning Edhis d	Pakistan cricket team wore black armbands for Edhi…
10	2.3352	3/18/2016	WC T20 Waqar confident Pakistan will defeat Indi	Waqar Younis expressed confidence Pakistan will defeat India…

4. Discussion

Findings: The implementation confirms that BM25 is a robust baseline for small-to-medium text corpora. The decision to merge Headlines and Body text proved crucial; without it, queries specific to titles often returned lower relevance scores. The preprocessing step was the most time-consuming part of the pipeline, but it significantly reduced the index size by removing stopwords.I have used just BM25 approach as it is more better than the simple TF-IDF approach

Shortcomings: The primary limitation of the current system is the lack of **Semantic Understanding**. Since BM25 is a "Bag of Words" model, it cannot understand synonyms. For example, a search for "Car" might not return articles that only mention "Automobile" or "Vehicle."

Future Improvements:

1. Saving the model to improve the computation and just

5. References

I have just found a youtube video [link]

6. Disclosure of AI Use

I have used the Gemini AI tool

Prompt: "-Consider yourself as an senior software engineer

- I have uploaded the document read and understand the whole document

- which is best approach to implement (don't provide the code)

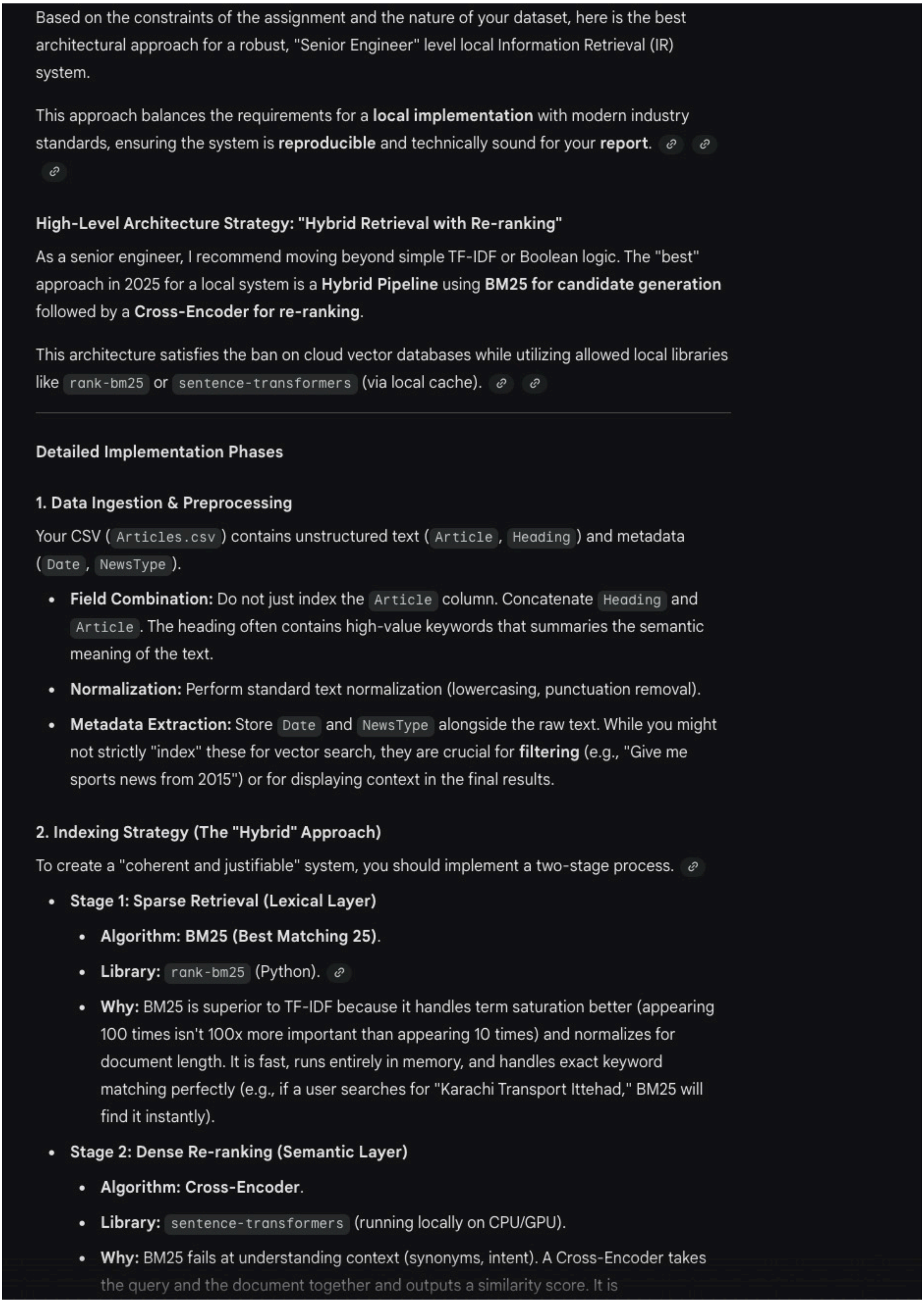
- The dataset that we are using is just one single CSV file named "Articles.csv" and it has data in the below format :

ArticleDateHeadingNewsTypeKARACHI: The Sindh government has decided to bring down public transport fares by 7 per cent due to massive reduction in petroleum product prices by the federal government, Geo News reported.Sources said reduction in fares will be applicable on public transport, rickshaw, taxi and other means of traveling.Meanwhile, Karachi Transport Ittehad (KTI) has refused to abide by the government decision.KTI President Irshad Bukhari said the commuters are charged the lowest fares in Karachi as compare to other parts of the country, adding that 80pc vehicles run on Compressed Natural Gas (CNG). Bukhari said Karachi transporters will cut fares when decrease in CNG prices will be made. 1/1/2015sindh govt decides to cut public transport fares by 7pc kti rejbusiness

"

AI-response :

Use the hybrid-approach (BM25 and cross encoder)



Based on the constraints of the assignment and the nature of your dataset, here is the best architectural approach for a robust, "Senior Engineer" level local Information Retrieval (IR) system.

This approach balances the requirements for a **local implementation** with modern industry standards, ensuring the system is **reproducible** and technically sound for your **report**.

High-Level Architecture Strategy: "Hybrid Retrieval with Re-ranking"

As a senior engineer, I recommend moving beyond simple TF-IDF or Boolean logic. The "best" approach in 2025 for a local system is a **Hybrid Pipeline** using **BM25 for candidate generation** followed by a **Cross-Encoder for re-ranking**.

This architecture satisfies the ban on cloud vector databases while utilizing allowed local libraries like `rank-bm25` or `sentence-transformers` (via local cache).

Detailed Implementation Phases

1. Data Ingestion & Preprocessing

Your CSV (`Articles.csv`) contains unstructured text (`Article`, `Heading`) and metadata (`Date`, `NewsType`).

- **Field Combination:** Do not just index the `Article` column. Concatenate `Heading` and `Article`. The heading often contains high-value keywords that summaries the semantic meaning of the text.
- **Normalization:** Perform standard text normalization (lowercasing, punctuation removal).
- **Metadata Extraction:** Store `Date` and `NewsType` alongside the raw text. While you might not strictly "index" these for vector search, they are crucial for **filtering** (e.g., "Give me sports news from 2015") or for displaying context in the final results.

2. Indexing Strategy (The "Hybrid" Approach)

To create a "coherent and justifiable" system, you should implement a two-stage process.

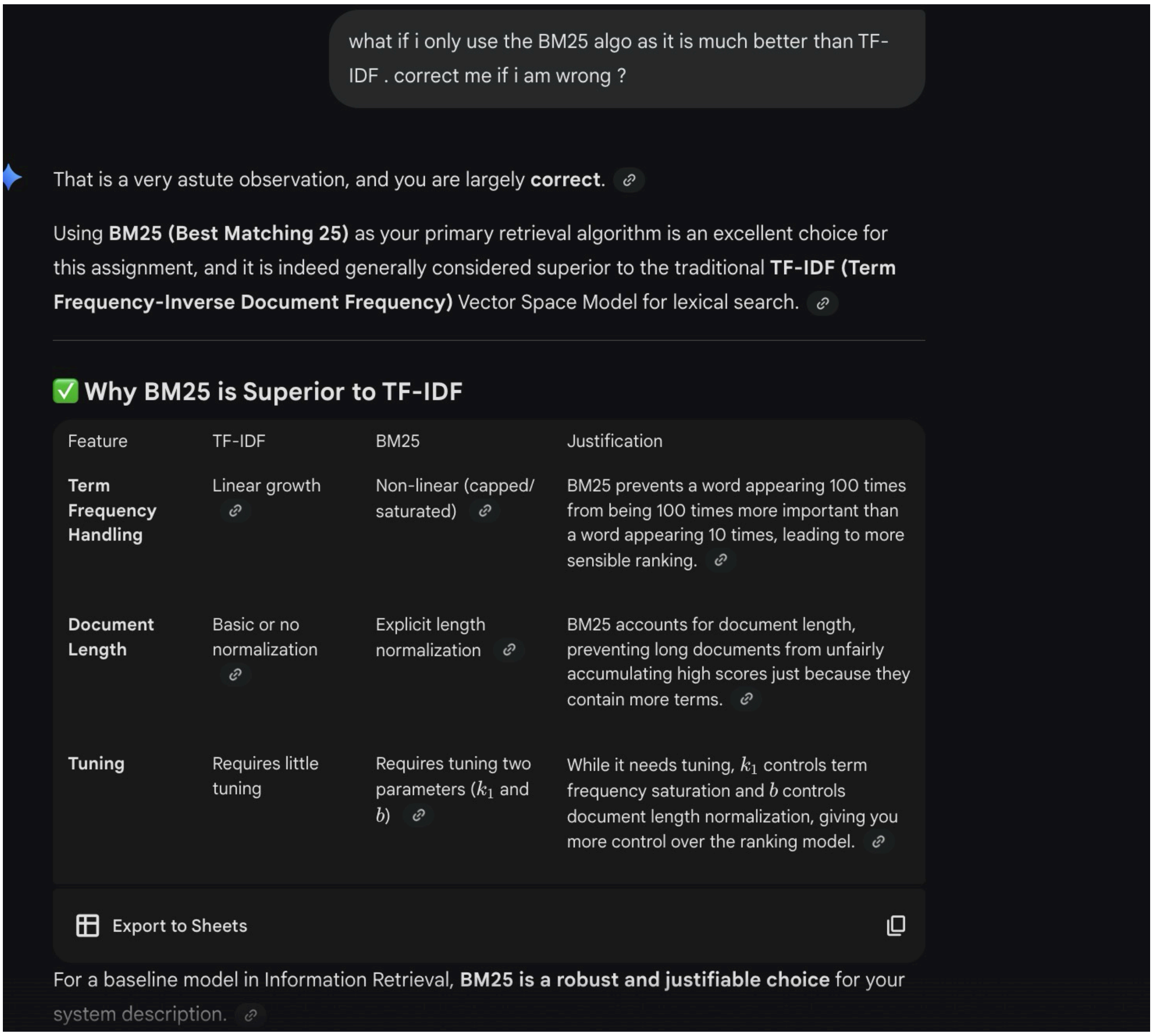
- **Stage 1: Sparse Retrieval (Lexical Layer)**
  - **Algorithm: BM25 (Best Matching 25)**.
  - **Library:** `rank-bm25` (Python).
  - **Why:** BM25 is superior to TF-IDF because it handles term saturation better (appearing 100 times isn't 100x more important than appearing 10 times) and normalizes for document length. It is fast, runs entirely in memory, and handles exact keyword matching perfectly (e.g., if a user searches for "Karachi Transport Ittehad," BM25 will find it instantly).
- **Stage 2: Dense Re-ranking (Semantic Layer)**
  - **Algorithm: Cross-Encoder**.
  - **Library:** `sentence-transformers` (running locally on CPU/GPU).
  - **Why:** BM25 fails at understanding context (synonyms, intent). A Cross-Encoder takes the query and the document together and outputs a similarity score. It is

Prompt : “what if i only use the BM25 algo as it is much better than TF-IDF . correct me if i am wrong ?

AI-response :

That is a very astute observation, and you are largely **correct**.

Using **BM25 (Best Matching 25)** as your primary retrieval algorithm is an excellent choice for this assignment, and it is indeed generally considered superior to the traditional **TF-IDF (Term Frequency-Inverse Document Frequency)** Vector Space Model for lexical searc



what if i only use the BM25 algo as it is much better than TF-IDF . correct me if i am wrong ?

That is a very astute observation, and you are largely **correct**.

Using **BM25 (Best Matching 25)** as your primary retrieval algorithm is an excellent choice for this assignment, and it is indeed generally considered superior to the traditional **TF-IDF (Term Frequency-Inverse Document Frequency)** Vector Space Model for lexical search.

✅ Why BM25 is Superior to TF-IDF

Feature	TF-IDF	BM25	Justification
Term Frequency Handling	Linear growth	Non-linear (capped/ saturated)	BM25 prevents a word appearing 100 times from being 100 times more important than a word appearing 10 times, leading to more sensible ranking.
Document Length	Basic or no normalization	Explicit length normalization	BM25 accounts for document length, preventing long documents from unfairly accumulating high scores just because they contain more terms.
Tuning	Requires little tuning	Requires tuning two parameters (k_1 and b)	While it needs tuning, k_1 controls term frequency saturation and b controls document length normalization, giving you more control over the ranking model.

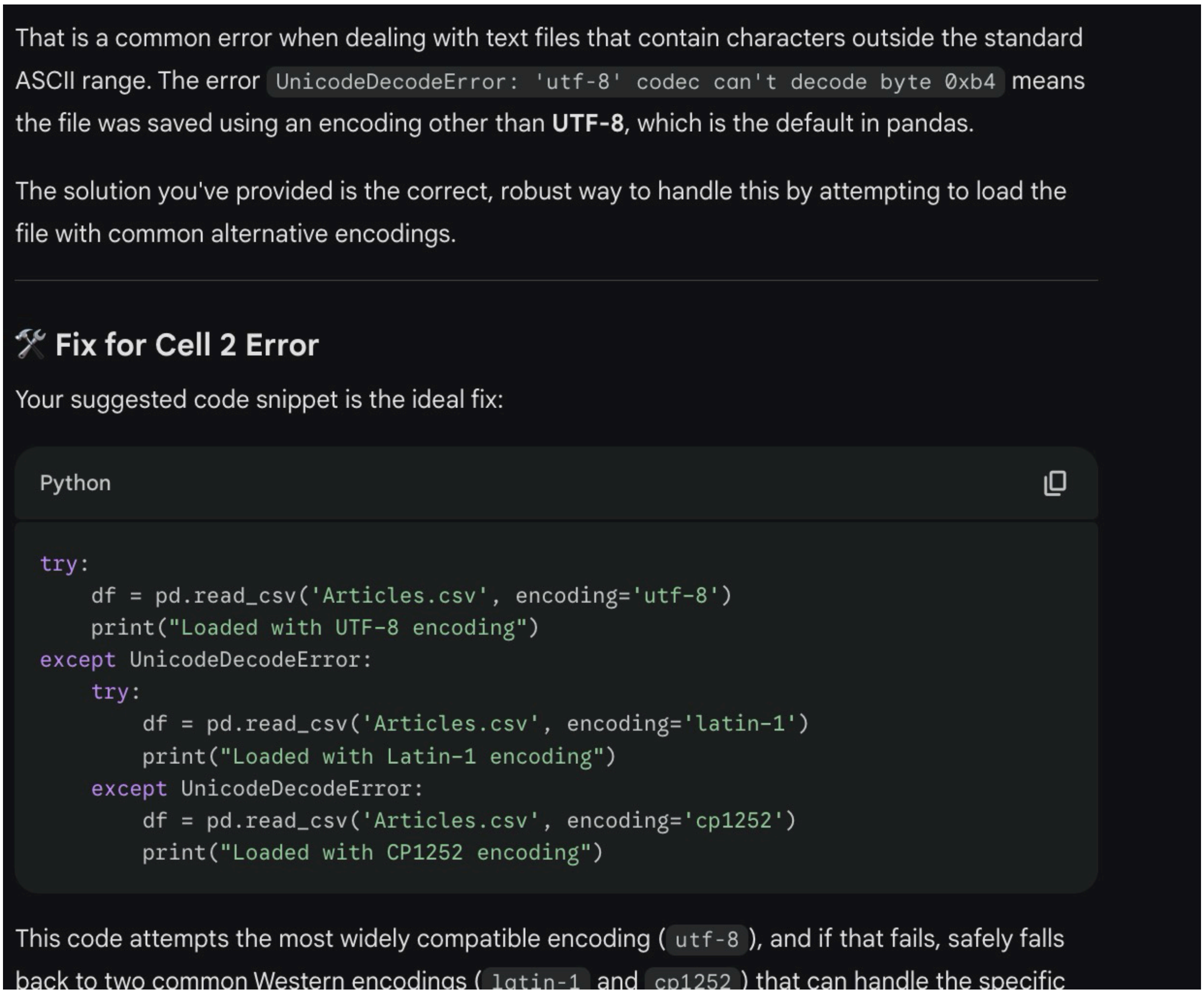
For a baseline model in Information Retrieval, **BM25 is a robust and justifiable choice** for your system description.

Prompt “I am getting the below error in cell 2 : ----> 2 df = pd.read_csv('Articles.csv') UnicodeDecodeError: 'utf-8' codec can't decode byte 0xb4 in position 1644: invalid start byteFix that error and also tell which changes I need to make in the furthur cells”

AI-response :

```
try:
    df = pd.read_csv('Articles.csv', encoding='utf-8')
    print("Loaded with UTF-8 encoding")
except UnicodeDecodeError:
    try:
        df = pd.read_csv('Articles.csv', encoding='latin-1')
        print("Loaded with Latin-1 encoding")
    except UnicodeDecodeError:
        df = pd.read_csv('Articles.csv', encoding='cp1252')
        print("Loaded with CP1252 encoding")
```

“The file is not opening in normal encoding so this code help me to verify that the file must be open in latin-1 encoding ”



That is a common error when dealing with text files that contain characters outside the standard ASCII range. The error `UnicodeDecodeError: 'utf-8' codec can't decode byte 0xb4` means the file was saved using an encoding other than **UTF-8**, which is the default in pandas.

The solution you've provided is the correct, robust way to handle this by attempting to load the file with common alternative encodings.

🛠️ Fix for Cell 2 Error

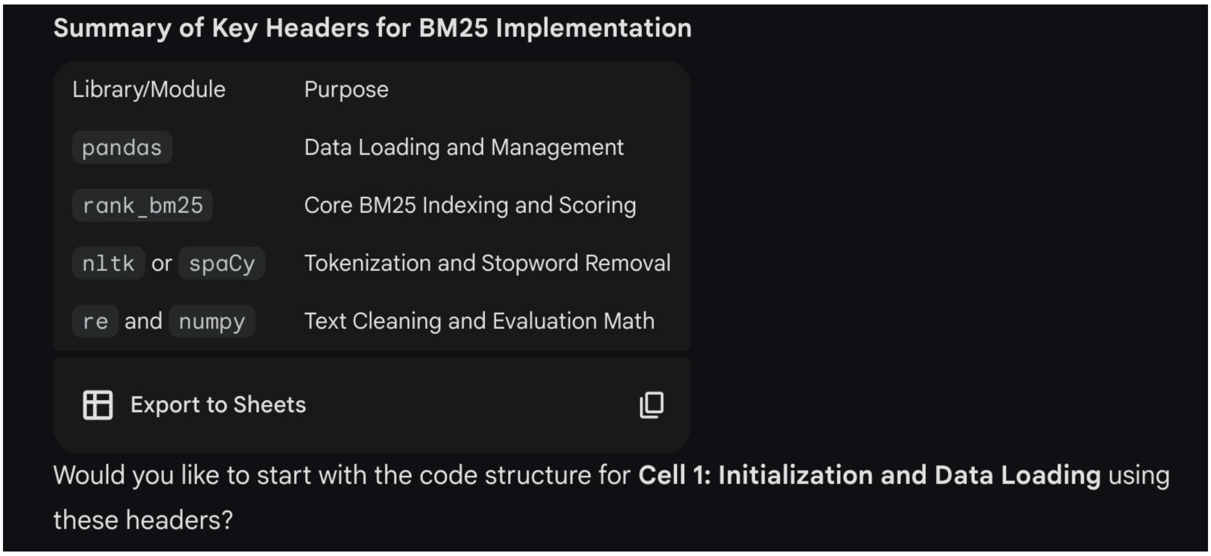
Your suggested code snippet is the ideal fix:

```python
try:
    df = pd.read_csv('Articles.csv', encoding='utf-8')
    print("Loaded with UTF-8 encoding")
except UnicodeDecodeError:
    try:
        df = pd.read_csv('Articles.csv', encoding='latin-1')
        print("Loaded with Latin-1 encoding")
    except UnicodeDecodeError:
        df = pd.read_csv('Articles.csv', encoding='cp1252')
        print("Loaded with CP1252 encoding")
```

This code attempts the most widely compatible encoding (`utf-8`), and if that fails, safely falls back to two common Western encodings (`latin-1` and `cp1252`) that can handle the specific

Prompt ; “which headers are needed for this project to me ”

AI-response “



Summary of Key Headers for BM25 Implementation

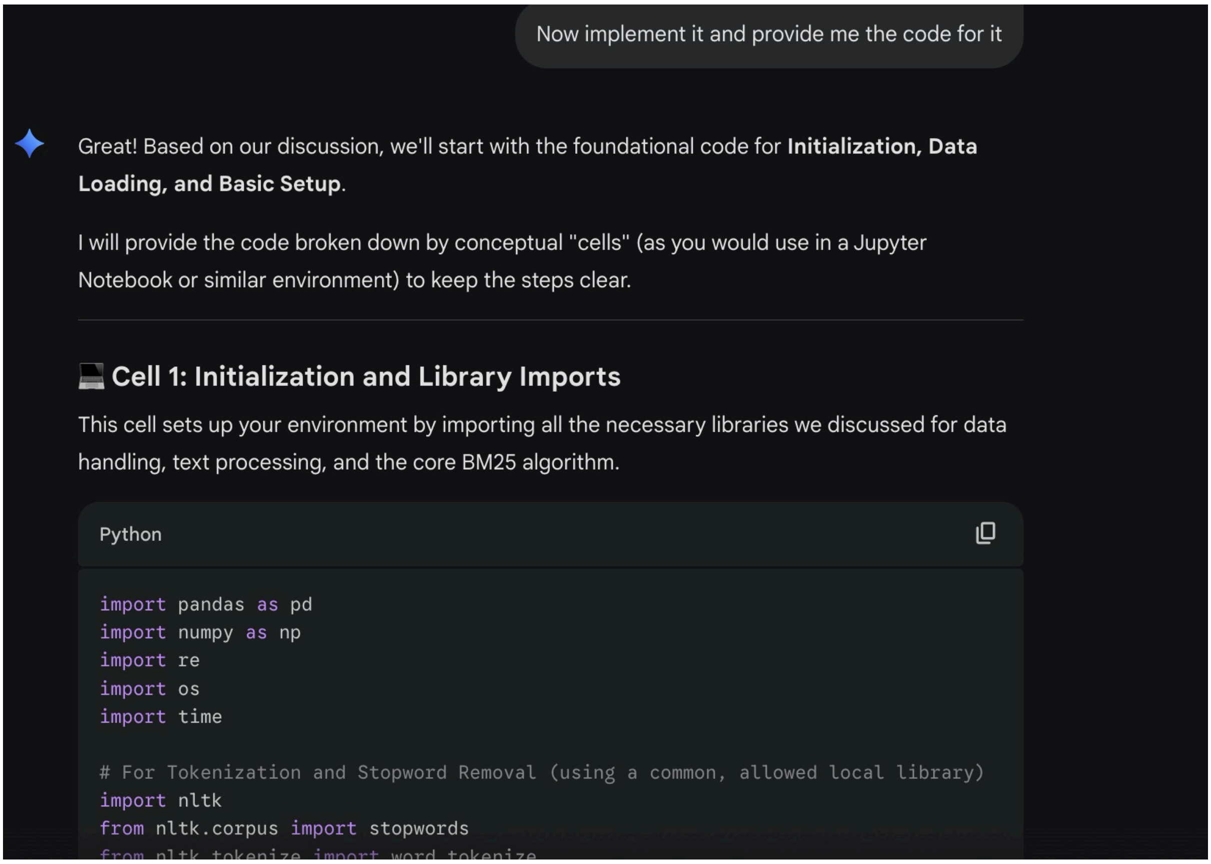
Library/Module	Purpose
`pandas`	Data Loading and Management
`rank_bm25`	Core BM25 Indexing and Scoring
`nltk` or `spaCy`	Tokenization and Stopword Removal
`re` and `numpy`	Text Cleaning and Evaluation Math

Would you like to start with the code structure for **Cell 1: Initialization and Data Loading** using these headers?

”

Prompt : “Now implement it and provide me the code for it**”**

AI-response :



Now implement it and provide me the code for it

Great! Based on our discussion, we'll start with the foundational code for **Initialization, Data Loading, and Basic Setup**.

I will provide the code broken down by conceptual "cells" (as you would use in a Jupyter Notebook or similar environment) to keep the steps clear.

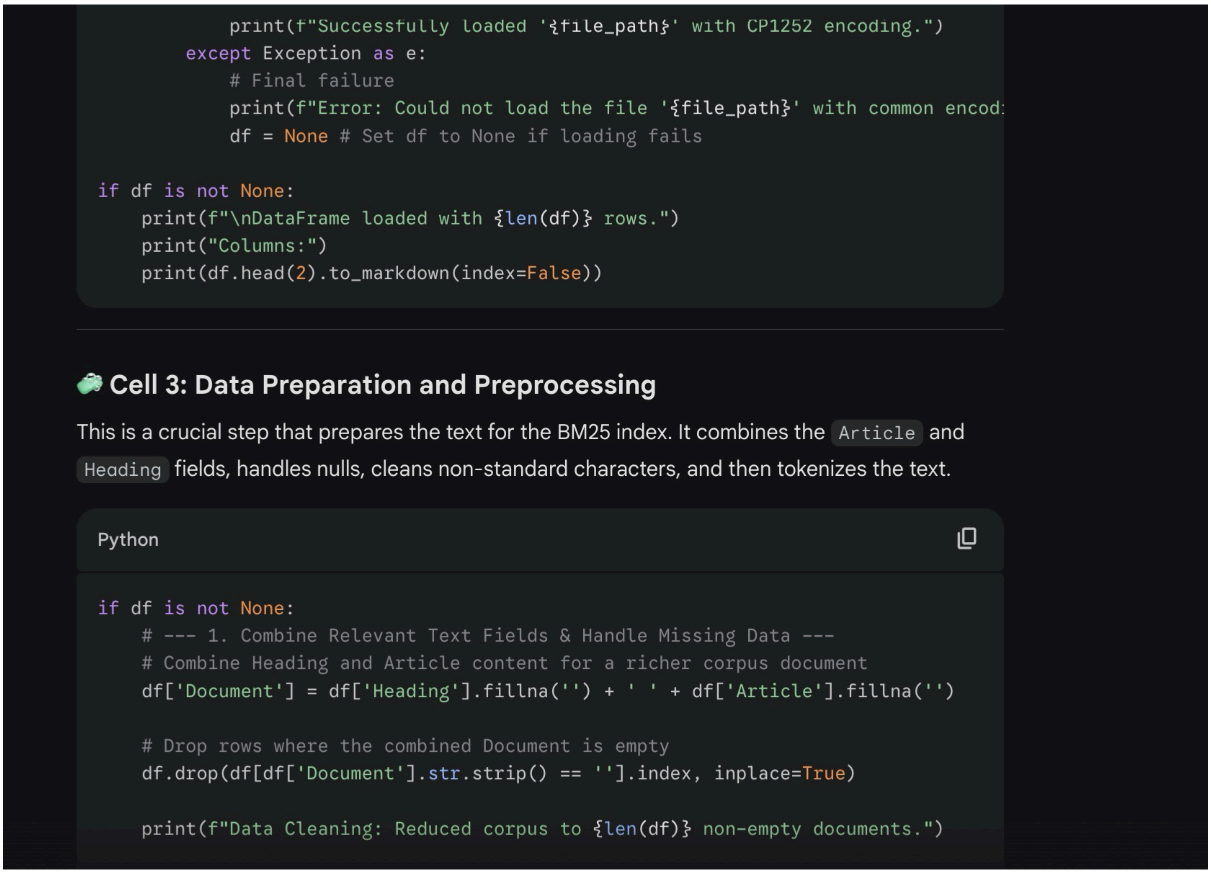
💻 Cell 1: Initialization and Library Imports

This cell sets up your environment by importing all the necessary libraries we discussed for data handling, text processing, and the core BM25 algorithm.

Python

```python
import pandas as pd
import numpy as np
import re
import os
import time

# For Tokenization and Stopword Removal (using a common, allowed local library)
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```



```python
            print(f"Successfully loaded '{file_path}' with CP1252 encoding.")
        except Exception as e:
            # Final failure
            print(f"Error: Could not load the file '{file_path}' with common encod
            df = None # Set df to None if loading fails

if df is not None:
    print(f"\nDataFrame loaded with {len(df)} rows.")
    print("Columns:")
    print(df.head(2).to_markdown(index=False))
```

🧼 Cell 3: Data Preparation and Preprocessing

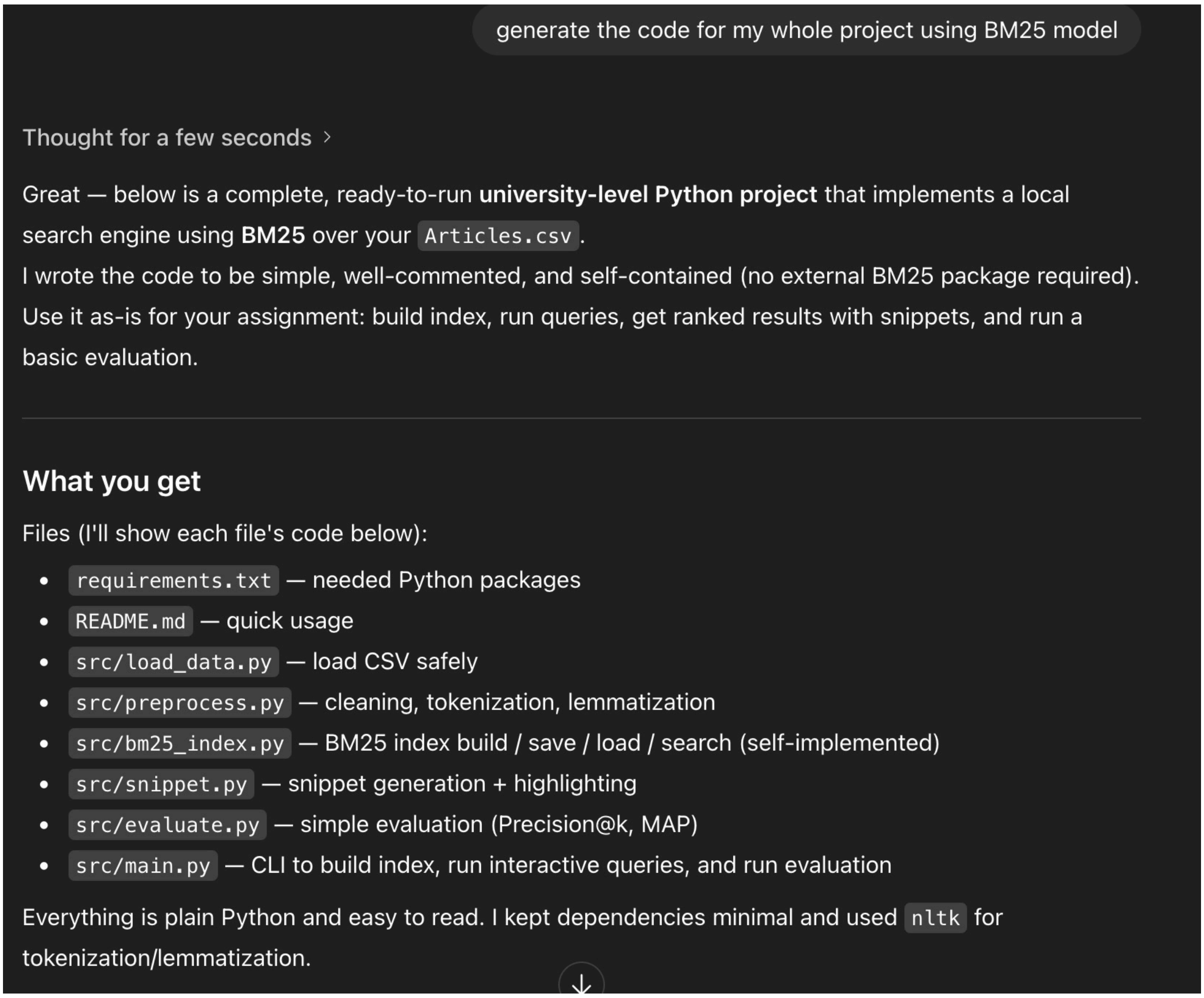
This is a crucial step that prepares the text for the BM25 index. It combines the `Article` and `Heading` fields, handles nulls, cleans non-standard characters, and then tokenizes the text.

Python

```python
if df is not None:
    # --- 1. Combine Relevant Text Fields & Handle Missing Data ---
    # Combine Heading and Article content for a richer corpus document
    df['Document'] = df['Heading'].fillna('') + ' ' + df['Article'].fillna('')

    # Drop rows where the combined Document is empty
    df.drop(df[df['Document'].str.strip() == ''].index, inplace=True)

    print(f"Data Cleaning: Reduced corpus to {len(df)} non-empty documents.")
```



generate the code for my whole project using BM25 model

Thought for a few seconds ›

Great — below is a complete, ready-to-run **university-level Python project** that implements a local search engine using **BM25** over your `Articles.csv`.
I wrote the code to be simple, well-commented, and self-contained (no external BM25 package required).
Use it as-is for your assignment: build index, run queries, get ranked results with snippets, and run a basic evaluation.

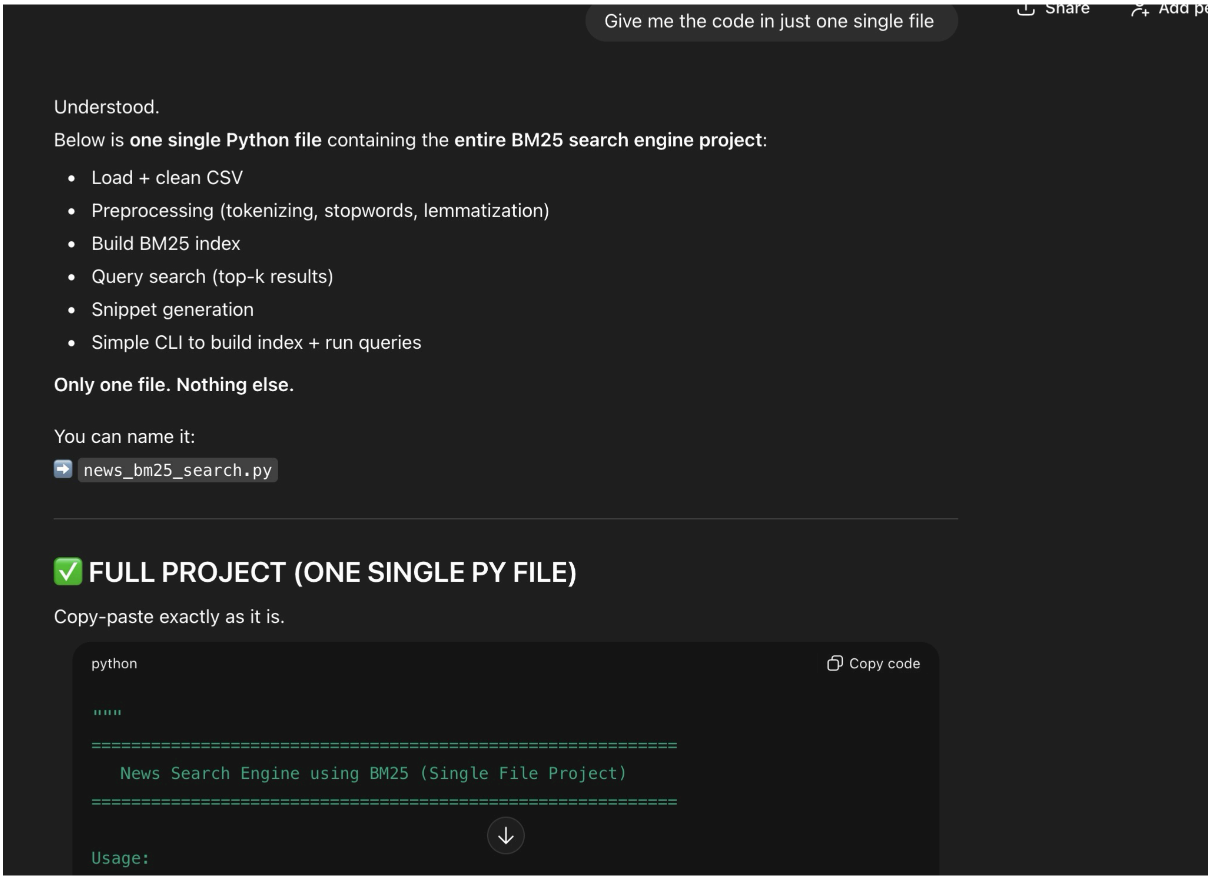
What you get

Files (I'll show each file's code below):

- `requirements.txt` — needed Python packages
- `README.md` — quick usage
- `src/load_data.py` — load CSV safely
- `src/preprocess.py` — cleaning, tokenization, lemmatization
- `src/bm25_index.py` — BM25 index build / save / load / search (self-implemented)
- `src/snippet.py` — snippet generation + highlighting
- `src/evaluate.py` — simple evaluation (Precision@k, MAP)
- `src/main.py` — CLI to build index, run interactive queries, and run evaluation

Everything is plain Python and easy to read. I kept dependencies minimal and used `nltk` for tokenization/lemmatization.

It has given me full structural code like in multiple files but then tell him to convert into single file as i don’t need code in multiple files



Give me the code in just one single file

Understood.

Below is **one single Python file** containing the **entire BM25 search engine project**:

- Load + clean CSV
- Preprocessing (tokenizing, stopwords, lemmatization)
- Build BM25 index
- Query search (top-k results)
- Snippet generation
- Simple CLI to build index + run queries

Only one file. Nothing else.

You can name it:

➡️ `news_bm25_search.py`

✅ FULL PROJECT (ONE SINGLE PY FILE)

Copy-paste exactly as it is.

```python
"""
===========================================================
   News Search Engine using BM25 (Single File Project)
===========================================================

Usage:
```

“I have created the code from AI from Gemini and GPT and taken the inspiration for him and then write my own code ”

“I have used AI for more reasons like understanding the function arguments and which functions to use [basic info] rather than google them I use AI for basic things ”

[Repo-link]