Keyword Based Search Engine

By:

Mohammed Farhan Ahmed

(Intern ID: IN1241283)

Submitted to Innomatics Research Labs,

Hyderabad,Telangana

Submission Date:

24/04/2024, 5PM

Project relation:

GenAI

**Overview**: in the 22nd century of technology, the internet has became a cluttered place, of something called data, data (in simple words) is a piece of information which can be found anywhere on the internet, the letters, the programs, etc are all forms of data, however because of its vast size of never ending supply, it is very difficult to organize each and every piece of data.

So, what does this have to do with the project ?, well in today’s era searching for subtitle files has become very difficult that there is a vast increase in SEO keywords for websites, so today we will be creating a search engine for individuals in order for them to find the correct srt (Subtitle files), using TFIDF-Cosine similarity techniques.

**Problem Statement:** Develop an advanced search engine algorithm that efficiently retrieves subtitles based on user queries, with a specific emphasis on subtitle content. The primary goal is to leverage natural language processing and machine learning techniques to enhance the relevance and accuracy of search results.

There are two types of Search Engines we can create:

* **Keyword Based**
* **Semantic Based**

Keyword Based Engine generally works on certain key words, for eg: in the search engine words such as “God”, “Smoke”, are generally used for input and in return their respective files are presented.

Conversely, Semantic Based is a more generalized version of Keyword Based; this takes user query *sentences* as input but requires a little more complex approach to built it, sentences such as “It’s time to party”, “Go forth” are used as input.

**Core Approach:**

1. Preprocessing of data:

* If you have limited compute resources, you can take a random 30% of the data.
* Clean: A possible cleaning step can be to remove time-stamps
* Note: Cleaning the Text data is crucial before vectorization

1. Vectorize the given Subtitle Documents

* Take the user query and vectorize the User Query.
* Cosine Similarity Calculation:
* Compute the cosine similarity between the vector of the documents and the vector of the user query.
* This similarity score determines the relevance of the documents to the user's

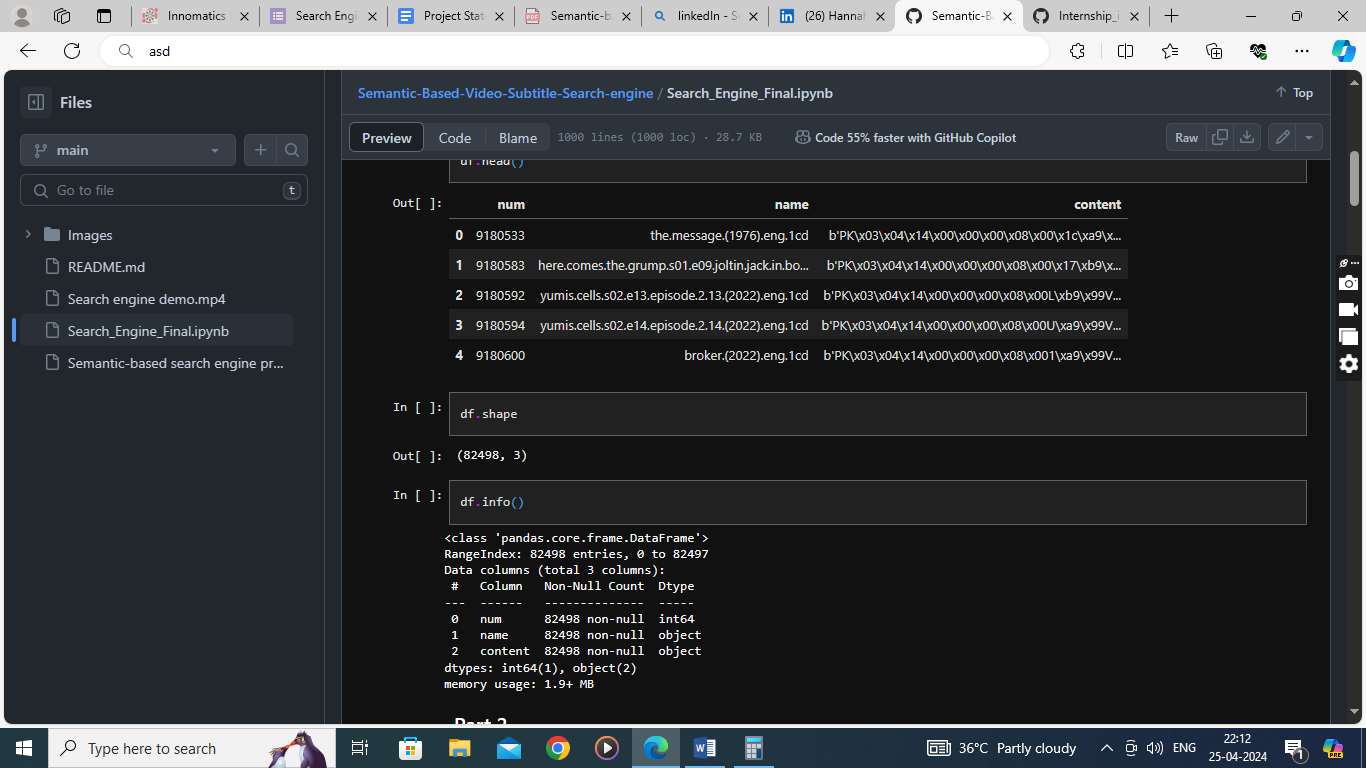
query.

1. Return the most similar documents

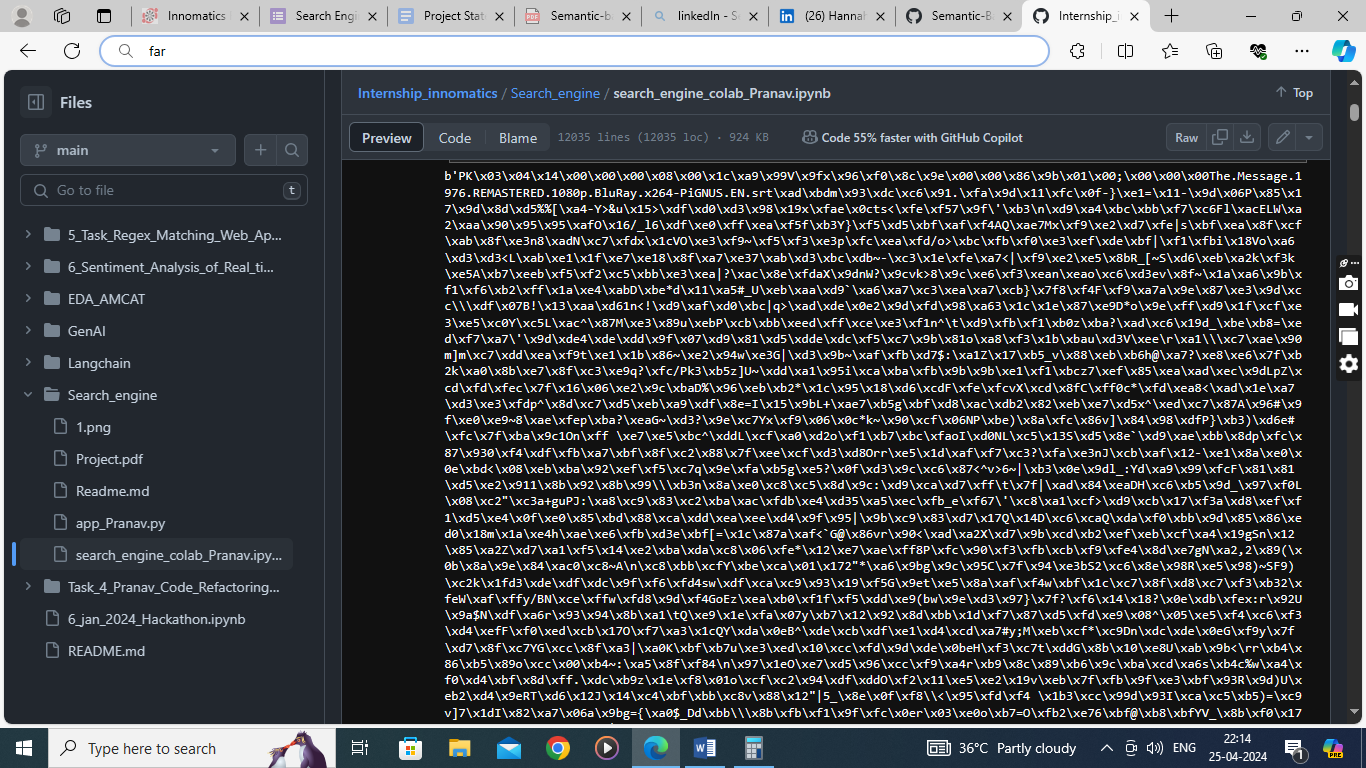
Steps sound simple but, how are we implementing them ?? Lets find out more….

1. Preprocessing of data:

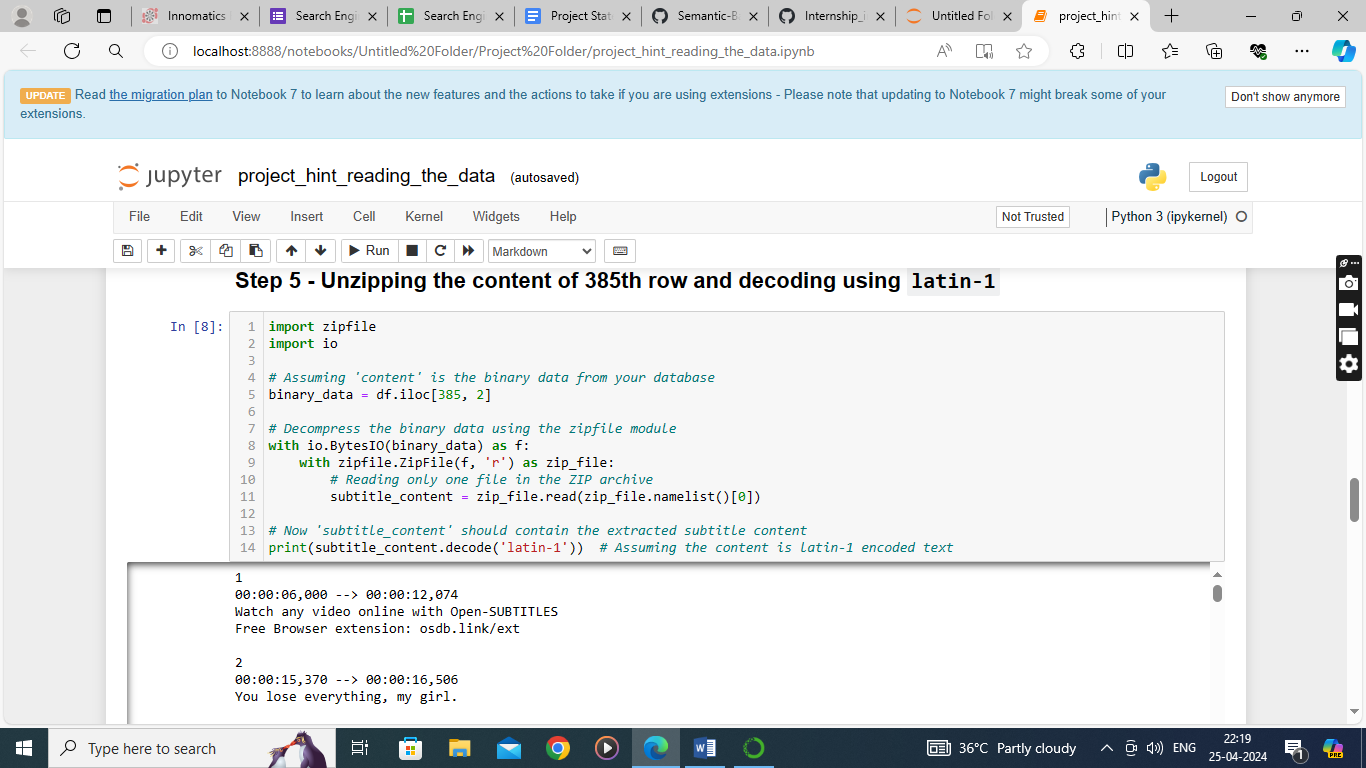
We are given the choice of preprocessing 30% of the original data, i.e out of 82000 files we can perform the project on atleast 24,600 files. So if we read the database given it will be something like this :

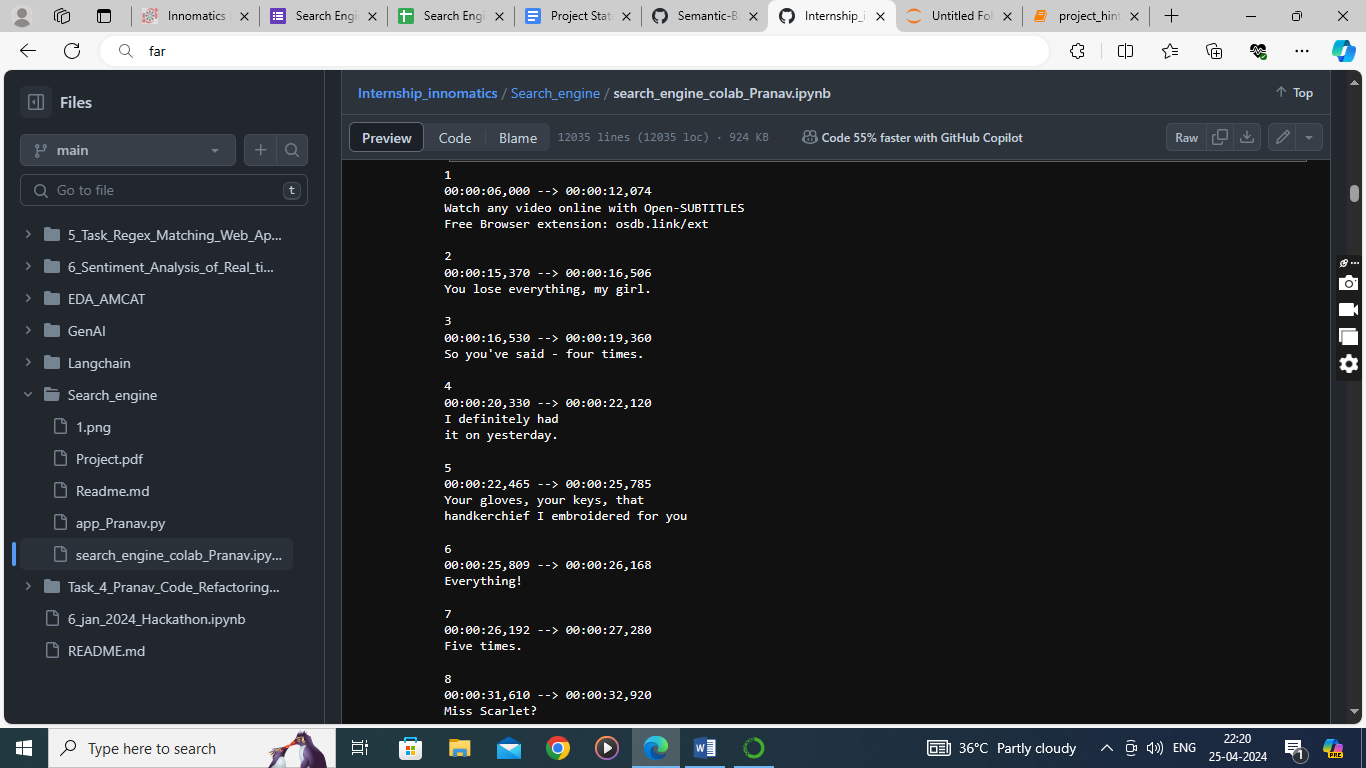


So, given the dataset, we have the name; its unique id and content….the content does not look readable, that is because it has been encrypted in latin code…that is, it is computer readable but not human readable. If we access one of the files it will be something like this.



We need to decrypt this in order to find out what kind of data are we dealing with,

So applying this code, we get :



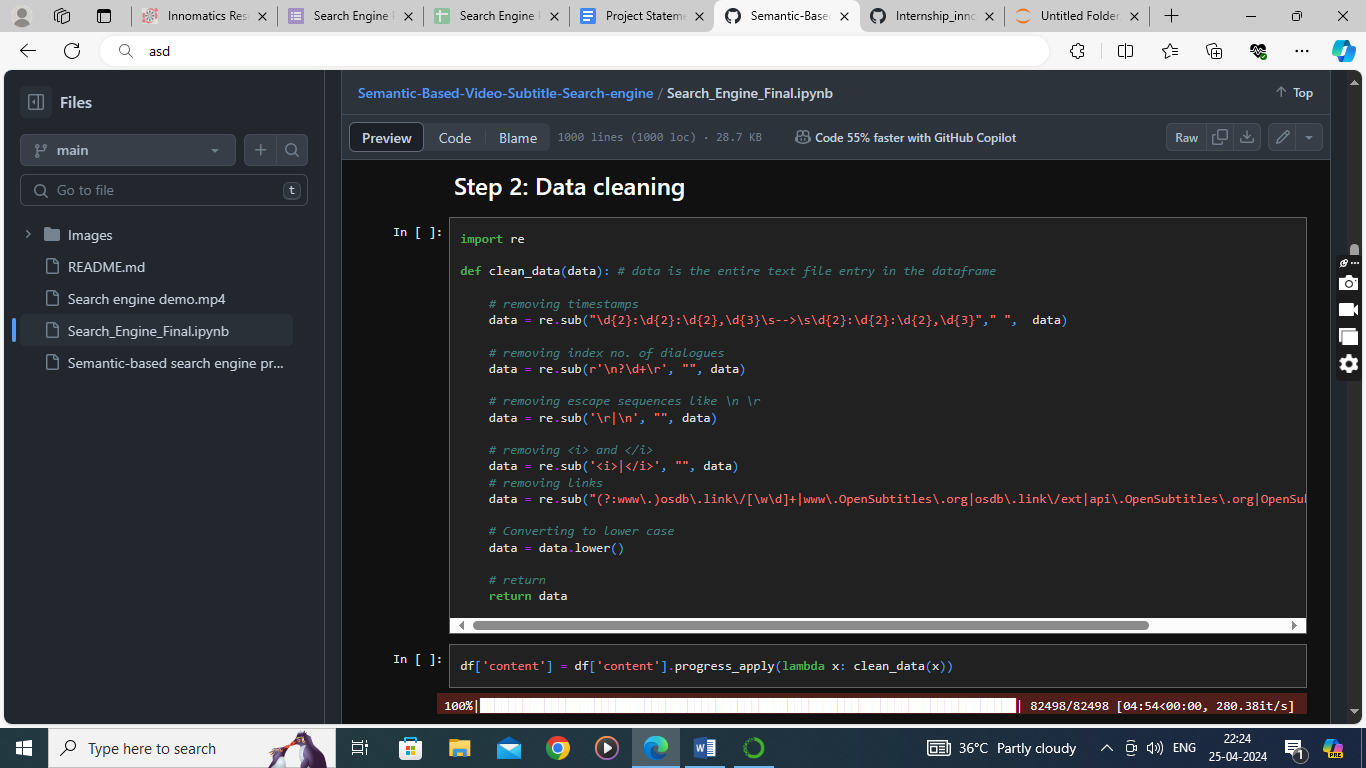
After applying the decryption process, we get the noise

data, as you can see it is unprocessed, dirty data

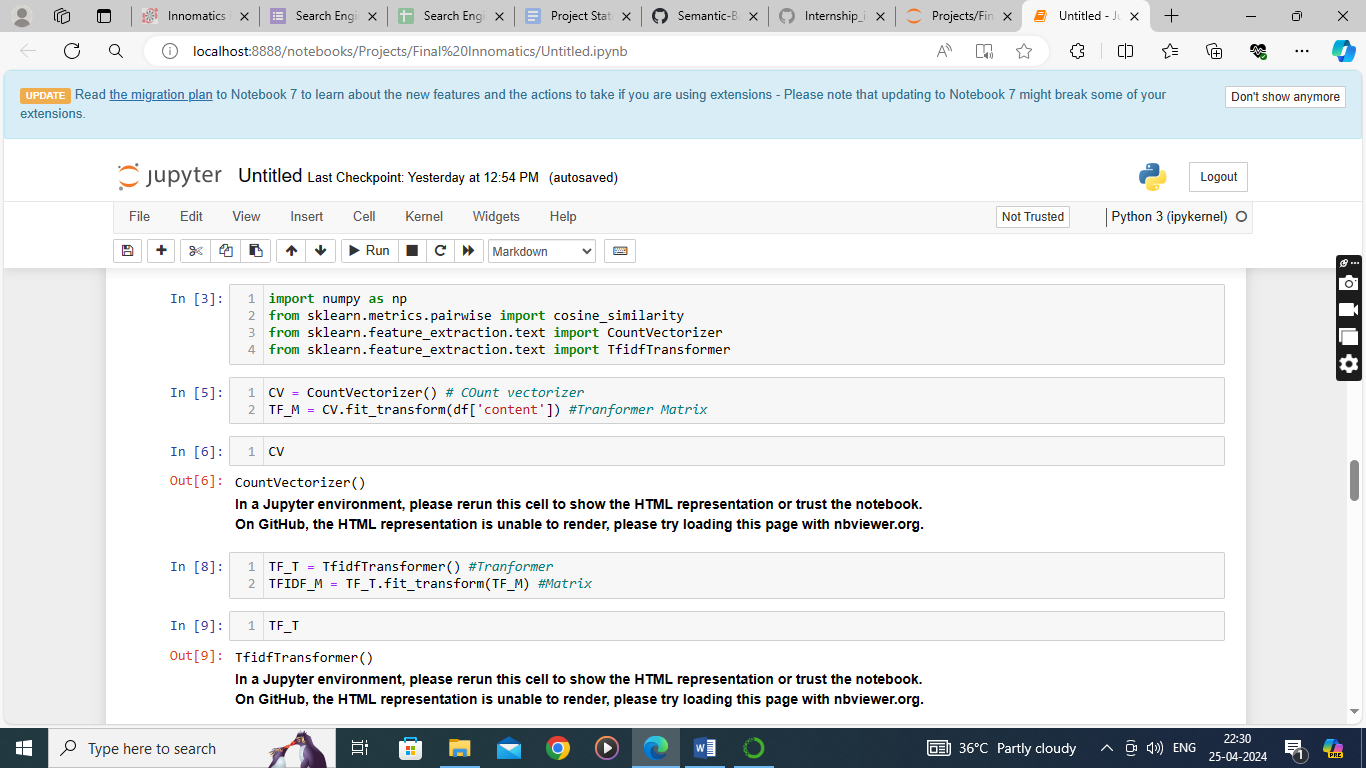
For the vectorization process only pure text (string based) data should be used for accurate calculation of matching similarities scores, i.e we need clean text data to get accurate scores.

We now apply the cleaning technique through a function like this :

* Removing timestamps
* Removing special characters
* Removing numbers
* Removing noisy sentences



After cleaning the data, we now apply the two techniques, CountVectorizer and TFIDF processes like this :



A little information about Count Vectorizer :

Count Vectorizer is a text processing technique used in natural language processing

(NLP) to convert a collection of text documents into a matrix of token counts. Each

document is represented by a vector where each element corresponds to the frequency

of a specific word in the document.

Eg: "The cat sat on the mat."

| and |brightly | cat | dog | in | is | mat | on | played | sat | shining | sun | the | yard |

| Doc1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 2 | 0 |

* Doc row represents a document.
* Each column represents a unique word in the corpus.
* The values represent the count of each word in the respective document.

A little information about TF-IDF:

TF-IDF stands for Term Frequency-Inverse Document Frequency. It's a numerical

statistic that reflects the importance of a word in a document relative to a collection of

documents (corpus). TF-IDF is commonly used in natural language processing and

information retrieval to determine the significance of a word in a document.

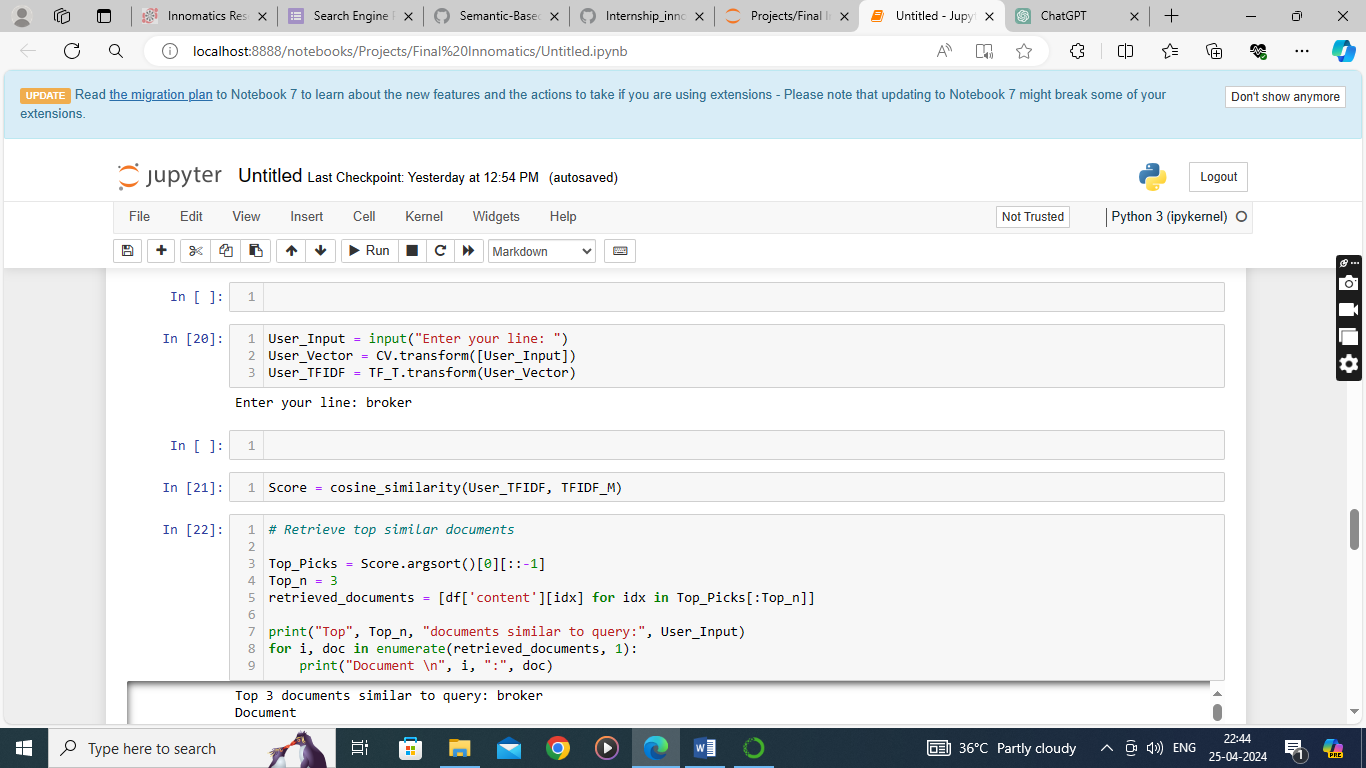
Term Frequency (TF) measures how frequently a term (word) appears in a document.

Inverse Document Frequency (IDF) measures the importance of a term across the

entire corpus.

Now that we have a clear idea of what we are doing, let use now proceed to making a

user query input like this :



So, we created an input query,

Count vectorized it,

Transformed it using a sprase matrix,

calculated the cosine similarity between

those two.

Retrieved the srt files based on the cosine angle.

* Usually, we do have another step we are taking the bert approach, you would

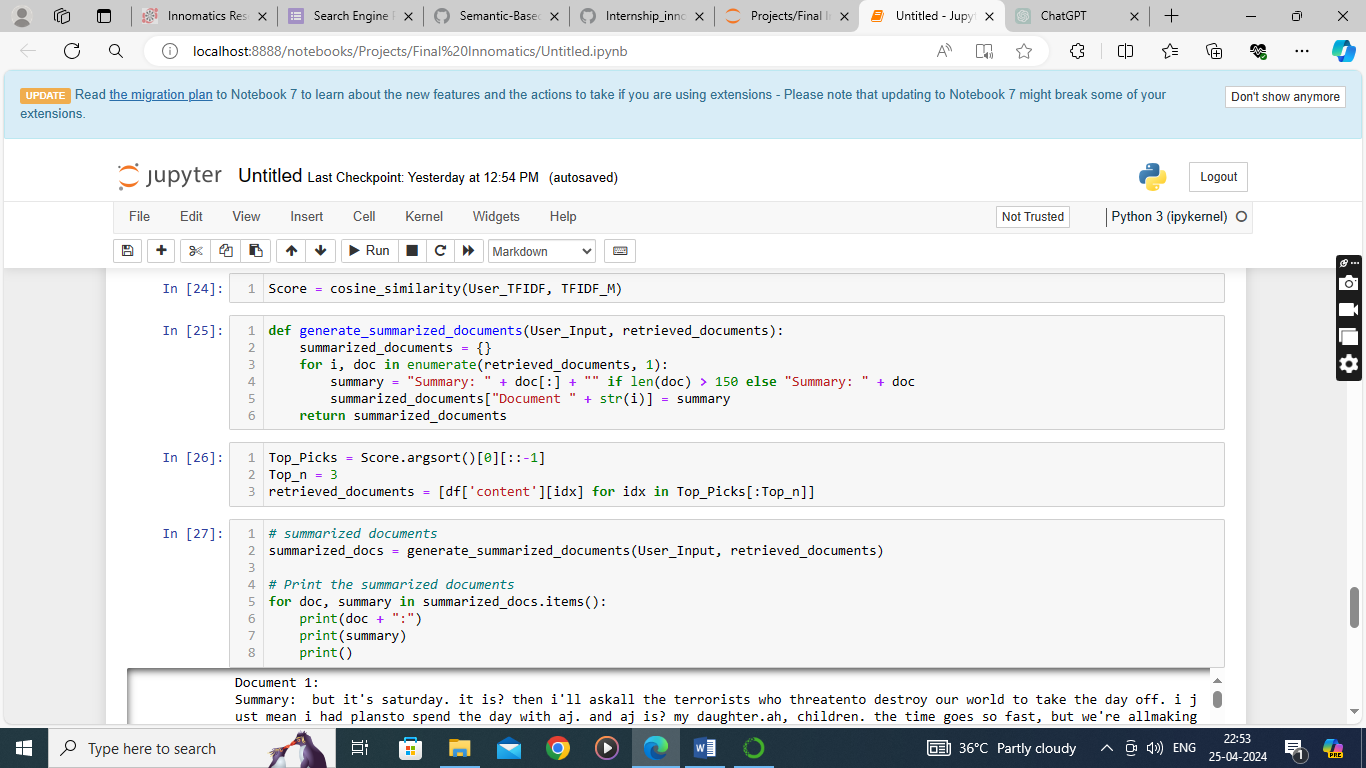
Have to use a transformer model, either using google (Gemini) or Openai (Chatgpt) to break them into chunks (small pieces of a big document), vectorize and store them in chromaDB. But we are making a keyword based search engine.

* The drawback of Semantic search engine is computational resources, a minimum of 16 GB ram is required in the software usage, for a transformer to run.

Now that our project is running we just need to make a function where instead of

retrieving the whole document, we just retrieve only 5-10 lines of the content and that

way, we can retrieve the doc using keyword engine, using this code:



Lastly this is the linkedIn post where the video demonstration is shown using Flask

Framework:

<https://www.linkedin.com/posts/mohammed-farhan-041a88238_disclaimer-this-post->

is-brought-to-you-activity-7189316844187496450-HcP6?utm\_source=share&utm\_medium=member\_desktop