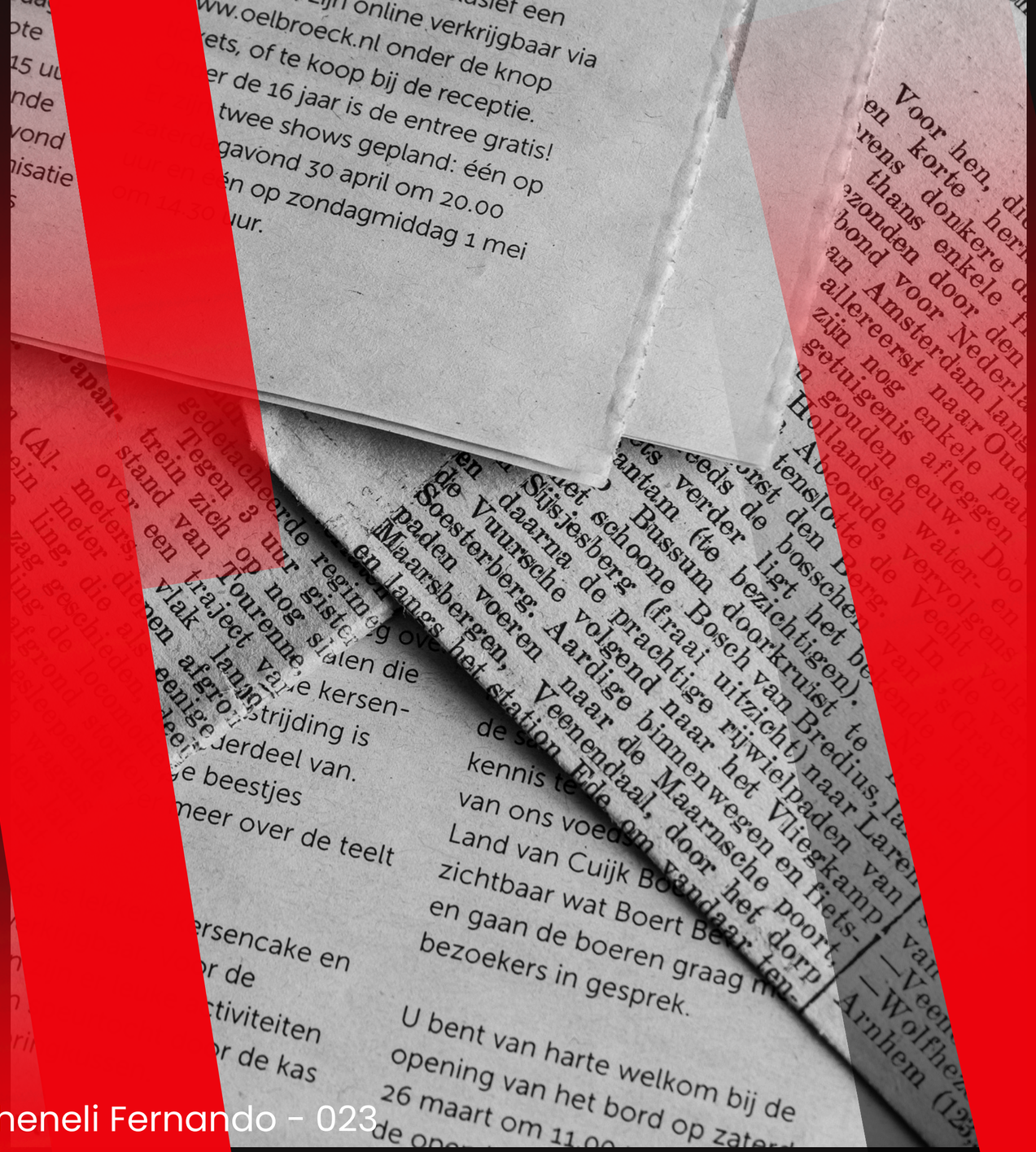
RETRIEVO

Dive Into The News That Matters



Oketha Senadeera - 001 Sadeeka Dissanayaka - 018 Sheneli Fernando - 023

OVERVIEW

- **Objective:** Retrieve relevant news articles based on a user query
- **Models Used**:
  - Vector Space Model (VSM)
  - Query Likelihood Model (QLM)
- **Tech Stack**: Python, Streamlit, Pandas
- **Final Output**: Interactive web app for model comparison

WORKFLOW

User Inputs Query

User Sets Parameters (Number Of Results | Lambda for QLM)

Query is Tokenized

Model is Selected (VSM | QLM)

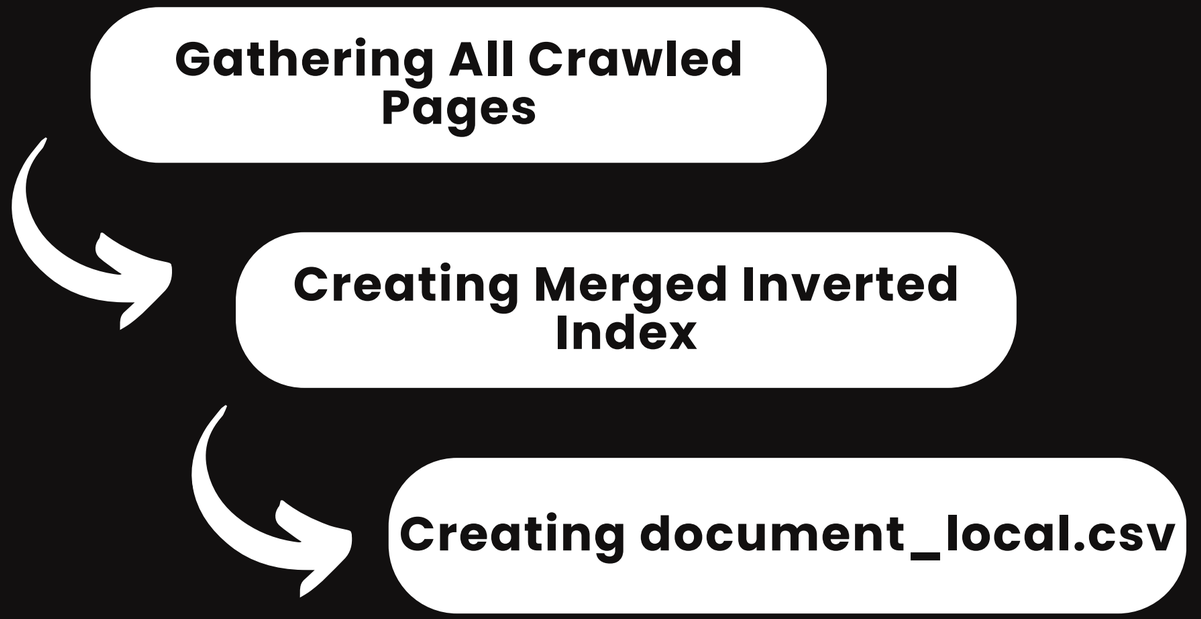
Sort Results by Score

Display Results

SEED PAGES

BBC	CNN	Al-Jazeera

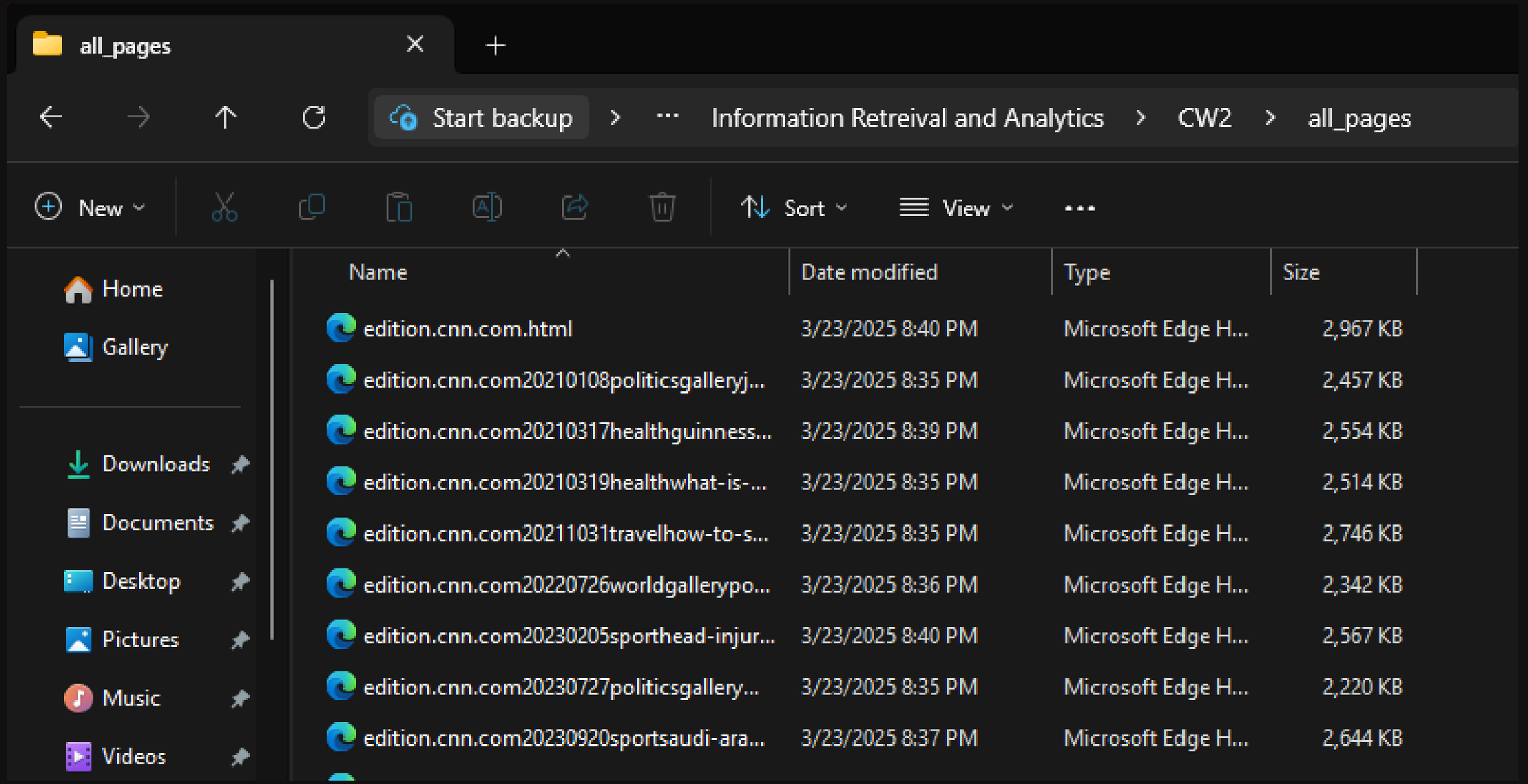
PREPARATION



PREPERATION

all_pages folder

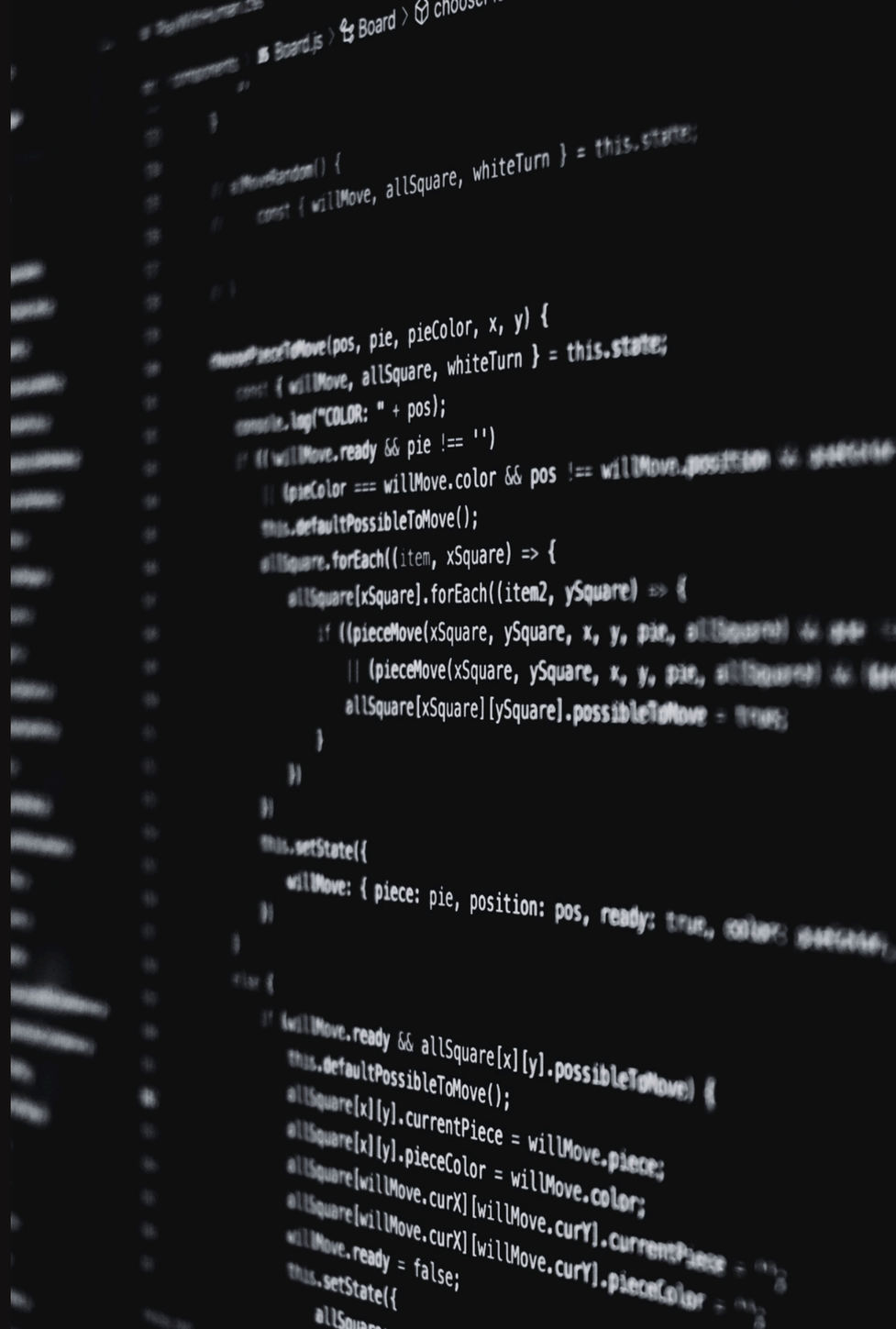
- Gathering all the crawled pages into all_pages folder.
- This folder contains all the raw HTML documents
- The actual content users might search for


all_pages
Start backup
Information Retreival and Analytics
CW2
all_pages
New
Sort
View
Home
Gallery
Downloads
Documents
Desktop
Pictures
Music
Videos
Name
Date modified
Type
Size
edition.cnn.com.html 3/23/2025 8:40 PM Microsoft Edge H... 2,967 KB
edition.cnn.com20210108politicsgalleryj... 3/23/2025 8:35 PM Microsoft Edge H... 2,457 KB
edition.cnn.com20210317healthguinness... 3/23/2025 8:39 PM Microsoft Edge H... 2,554 KB
edition.cnn.com20210319healthwhat-is-... 3/23/2025 8:35 PM Microsoft Edge H... 2,514 KB
edition.cnn.com20211031travelhow-to-s... 3/23/2025 8:35 PM Microsoft Edge H... 2,746 KB
edition.cnn.com20220726worldgallerypo... 3/23/2025 8:36 PM Microsoft Edge H... 2,342 KB
edition.cnn.com20230205sporthead-injur... 3/23/2025 8:40 PM Microsoft Edge H... 2,567 KB
edition.cnn.com20230727politicsgallery... 3/23/2025 8:35 PM Microsoft Edge H... 2,220 KB
edition.cnn.com20230920sportsaudi-ara... 3/23/2025 8:37 PM Microsoft Edge H... 2,644 KB

PREPERATION

merged_inverted_index

- The inverted index maps each word (term) to the list of documents (docIDs) where that word appears.
- Makes searching very fast
- Mapped according to doc_id



```python
from bs4 import BeautifulSoup
import os
import re
import string
import json
import nltk
from collections import defaultdict, Counter
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import pandas as pd

nltk.download('stopwords') #filtering out common words like "the", "and"
nltk.download('punkt') #tokenizing words properly


# Directory for storing crawled pages and index file
OUTPUT_DIR = "all_pages"
INDEX_FILE = "merged_index.json"


# Initialize NLP tools
stemmer = PorterStemmer() # reduces words to their root form
stop_words = set(stopwords.words("english")) #Stopwords are removed because they don't add useful meaning
```

```python
def process_text(text):
    """Cleans, tokenizes, removes stopwords, and applies stemming."""
    text = text.lower()
    text = re.sub(r'\d+', '', text)  # Remove numbers
    text = text.translate(str.maketrans("", "", string.punctuation))  # Remove punctuation
    words = word_tokenize(text)  # Tokenize words
    words = [stemmer.stem(word) for word in words if word not in stop_words and len(word) > 2]
    return words
```

```python
def build_inverted_index(output_dir):
    """Builds an inverted index and maps document IDs to filenames."""
    inverted_index = defaultdict(set) #maps each word to a set of doc IDs
    doc_id_mapping = {}  # Store and maps each doc ID to its filename.
    doc_id = 0 # Initialize document ID counter

    for filename in sorted(os.listdir(output_dir)):
        if filename.endswith(".html"): #Associates current doc_id with its filename
            doc_id += 1
            file_path = os.path.join(output_dir, filename)
            doc_id_mapping[doc_id] = filename  # Store document ID mapping

            try: #Loops through all HTML files in all_pages
                with open(file_path, "r", encoding="utf-8") as file: #Opens and reads each HTML file
                    soup = BeautifulSoup(file, "html.parser") #Removes all HTML tags
                    text = soup.get_text() #Extracts clean text for indexing

                    # Skip empty documents
                    if not text.strip():
                        print(f"Warning: {filename} contains no extractable text.")
                        continue

                    words = process_text(text)

                    # Skip documents that result in no valid words
                    if not words:
                        print(f"Warning: {filename} resulted in no valid words.")
                        continue

                    for word in set(words):
                        inverted_index[word].add(doc_id)

            except Exception as e:
                print(f"Error processing {filename}: {e}")

    # Convert sets to lists for JSON storage
    for term in inverted_index:
        inverted_index[term] = list(inverted_index[term]) #Sets are not JSON serializable. Lists are. Needed to save the index as a .json file.
    return dict(inverted_index), doc_id_mapping
```

```python
def save_index_to_json(inverted_index, doc_id_mapping, output_file):
    """Saves the inverted index and document ID mapping to a JSON file."""
    index_data = {
        "inverted_index": inverted_index,
        "doc_id_mapping": doc_id_mapping
    }
    with open(output_file, 'w', encoding='utf-8') as file:
        json.dump(index_data, file, indent=4)

def load_index_from_json(input_file):
    """Loads the inverted index and document ID mapping from a JSON file."""
    with open(input_file, 'r', encoding='utf-8') as file:
        index_data = json.load(file)
    doc_id_mapping = {int(k): v for k, v in index_data['doc_id_mapping'].items()}
    return index_data['inverted_index'], doc_id_mapping

def main():
    print("Starting indexing process...")
    inverted_index, doc_id_mapping = build_inverted_index(OUTPUT_DIR)
    save_index_to_json(inverted_index, doc_id_mapping, INDEX_FILE)
    print(f"Indexing completed. Index saved to '{INDEX_FILE}'")

    # Load and verify index
    loaded_index, loaded_mapping = load_index_from_json(INDEX_FILE)
    print(f"Indexed {len(loaded_mapping)} documents")
    print(f"Created index with {len(loaded_index)} unique terms")

if __name__ == "__main__":
    main()
```

merged_index.json

```
{
    "inverted_index": {
        "widaknurphotogetti": [
            1,
            395,
            556,
            503
        ],
        "canyon": [
            576,
            1,
            585,
            1834,
            107,
            395,
            141,
            335,
            463,
            403,
            503,
            570,
            574
        ],
        "fifth": [
            1,
```

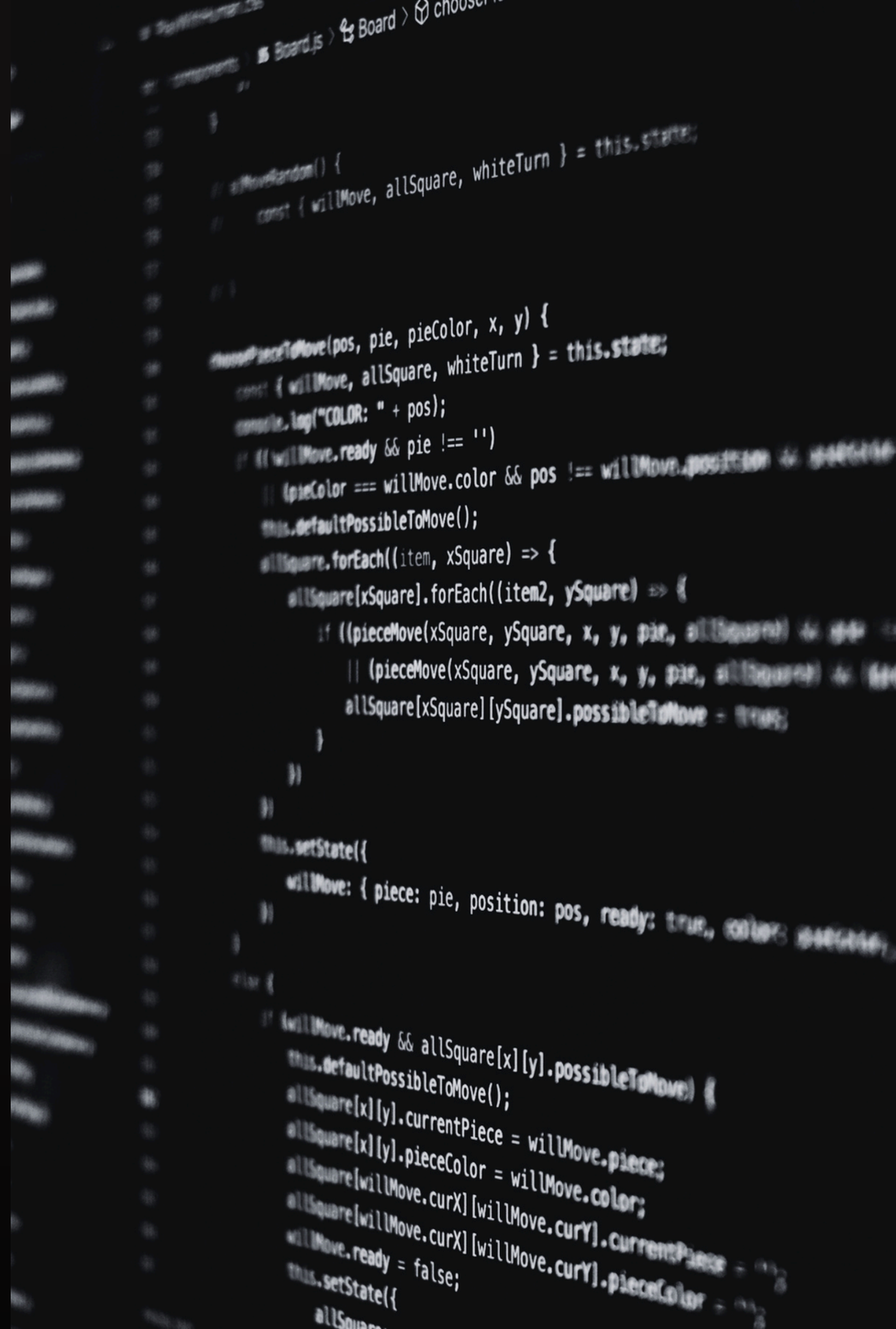
merged_index.json

```
            1860
        ],
        "taylforthphil": [
            1860
        ],
        "thirti": [
            1860
        ],
        "ethel\u00e2\u0080\u0099": [
            1860
        ]
    },
    "doc_id_mapping": {
        "1": "edition.cnn.com.html",
        "2": "edition.cnn.com20210108politicsgalleryjoe-bidenindex.html.html",
        "3": "edition.cnn.com20210317healthguinness-beer-good-for-you-wellnessindex.html.html",
        "4": "edition.cnn.com20210319healthwhat-is-nowruz-meaning-traditions-wellnessindex.html.html",
        "5": "edition.cnn.com20211031travelhow-to-survive-blizzard-weather-wellnessindex.html.html",
        "6": "edition.cnn.com20220726worldgallerypope-francisindex.html.html",
        "7": "edition.cnn.com20230205sporthead-injury-suicide-female-athletes-intl-spt-cmdindex.html.htm
        "8": "edition.cnn.com20230727politicsgallerymitch-mcconnellindex.html.html",
        "9": "edition.cnn.com20230920sportsaudi-arabia-soccer-spl-bin-salman-intl-spt-cmd-dgindex.html.h
        "10": "edition.cnn.com20231030sportenhanced-games-olympics-doping-spt-intlindex.html.html",
        "11": "edition.cnn.com20231122techopenai-altman-returns-hnk-intlindex.html.html",
        "12": "edition.cnn.com20231122techopenai-sam-altman-chaos-explained-intl-hnkindex.html.html",
        "13": "edition.cnn.com20240219sportpsilocybin-athletes-life-changing-injuries-psychedelics-spt-i
```

PREPERATION

document_local.csv

- The csv file containing the metadata for each document.
- Includes, doc_id, title, link (file_path)
- This CSV allows the search system to show meaningful results and clickable links to view the pages



```python
import json
import csv
from bs4 import BeautifulSoup
from pathlib import Path
import urllib.parse

# Configuration
INVERTED_INDEX_PATH = "merged_index.json"
HTML_FOLDER = "all_pages"
OUTPUT_CSV = "documents_local.csv"
LOCAL_SERVER_URL = "http://localhost:8000/"

# Load inverted index
with open(INVERTED_INDEX_PATH, "r", encoding="utf-8") as f:
    doc_id_mapping = json.load(f).get("doc_id_mapping", {})

# Create CSV with proper encoding
with open(OUTPUT_CSV, "w", newline="", encoding="utf-8") as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(["DocumentID", "Link", "LinkLocal", "Title"])

    for doc_id, filename in doc_id_mapping.items():
        file_path = Path(HTML_FOLDER) / filename

        if not file_path.exists():
            print(f"Missing file: {file_path}")
            continue

        # Extract title
        with open(file_path, "r", encoding="utf-8") as html_file:
            soup = BeautifulSoup(html_file, "html.parser")
            title = soup.title.text.strip() if soup.title else "No Title"

        # Create file URL
        abs_path = file_path.resolve()
        encoded_path = urllib.parse.quote(str(abs_path), safe=":/")
        encoded_path = encoded_path.replace("%5C", "/")  # Ensure forward slashes
        file_url = f"file:///{encoded_path}"

        # Create local server URL
        local_url = f"{LOCAL_SERVER_URL}{filename}"

        writer.writerow([doc_id, file_url, local_url, title])

print(f"Successfully generated {OUTPUT_CSV}")
```

```
Successfully generated documents_local.csv
```

DocumentID	Link	LinkLocal	Title
1	file:///C:/Users/oketh/OneDrive/Desktop/HNDDS/Information%20Retreival%20and	http://localhost:8000/edition.cnn.com.html	Breaking News, Latest News and Videos | CNN
2	file:///C:/Users/oketh/OneDrive/Desktop/HNDDS/Information%20Retreival%20and	http://localhost:8000/edition.cnn.com20210108politicsgallery	In pictures: Former President Joe Biden | CNN Poli
3	file:///C:/Users/oketh/OneDrive/Desktop/HNDDS/Information%20Retreival%20and	http://localhost:8000/edition.cnn.com20210317healthguinnes	Guinness beer: Is it really good for you? | CNN
4	file:///C:/Users/oketh/OneDrive/Desktop/HNDDS/Information%20Retreival%20and	http://localhost:8000/edition.cnn.com20210319healthwhat-is	What is Nowruz? Persian New Year traditions and
5	file:///C:/Users/oketh/OneDrive/Desktop/HNDDS/Information%20Retreival%20and	http://localhost:8000/edition.cnn.com20211031travelhow-to-s	How to survive in a blizzard | CNN
6	file:///C:/Users/oketh/OneDrive/Desktop/HNDDS/Information%20Retreival%20and	http://localhost:8000/edition.cnn.com20220726worldgalleryp	Pope Francis, in pictures | CNN
7	file:///C:/Users/oketh/OneDrive/Desktop/HNDDS/Information%20Retreival%20and	http://localhost:8000/edition.cnn.com20230205sporthead-inj	These young female athletes died by suicide. They
8	file:///C:/Users/oketh/OneDrive/Desktop/HNDDS/Information%20Retreival%20and	http://localhost:8000/edition.cnn.com20230727politicsgallery	Photos: Former GOP Senate leader Mitch McConne
9	file:///C:/Users/oketh/OneDrive/Desktop/HNDDS/Information%20Retreival%20and	http://localhost:8000/edition.cnn.com20230920sportsaudi-ar	Saudi Arabia is trying to disrupt soccerâ€™s world
10	file:///C:/Users/oketh/OneDrive/Desktop/HNDDS/Information%20Retreival%20and	http://localhost:8000/edition.cnn.com20231030sportenhance	A doping free-for-all Enhanced Games calls itself t
11	file:///C:/Users/oketh/OneDrive/Desktop/HNDDS/Information%20Retreival%20and	http://localhost:8000/edition.cnn.com20231122techopenai-al	Sam Altman returns to OpenAI in a bizarre reversal
12	file:///C:/Users/oketh/OneDrive/Desktop/HNDDS/Information%20Retreival%20and	http://localhost:8000/edition.cnn.com20231122techopenai-sa	OpenAIâ€™s wild week explained:Â How the Sam A
13	file:///C:/Users/oketh/OneDrive/Desktop/HNDDS/Information%20Retreival%20and	http://localhost:8000/edition.cnn.com20240219sportpsilocyb	These athletes suffered life-changing injuries. Ther
14	file:///C:/Users/oketh/OneDrive/Desktop/HNDDS/Information%20Retreival%20and	http://localhost:8000/edition.cnn.com20240301techaiscout-a	AiScout: Top soccer clubs are using a mobile app t
15	file:///C:/Users/oketh/OneDrive/Desktop/HNDDS/Information%20Retreival%20and	http://localhost:8000/edition.cnn.com20240316americasaxol	Why axolotls are everywhere â€” but a rarity in the

VECTOR SPACE MODEL

What is VSM?

- An Information Retrieval model that represents documents and queries as vectors.
- Relevance is determined by cosine similarity between the query vector and document vectors.

Key concepts

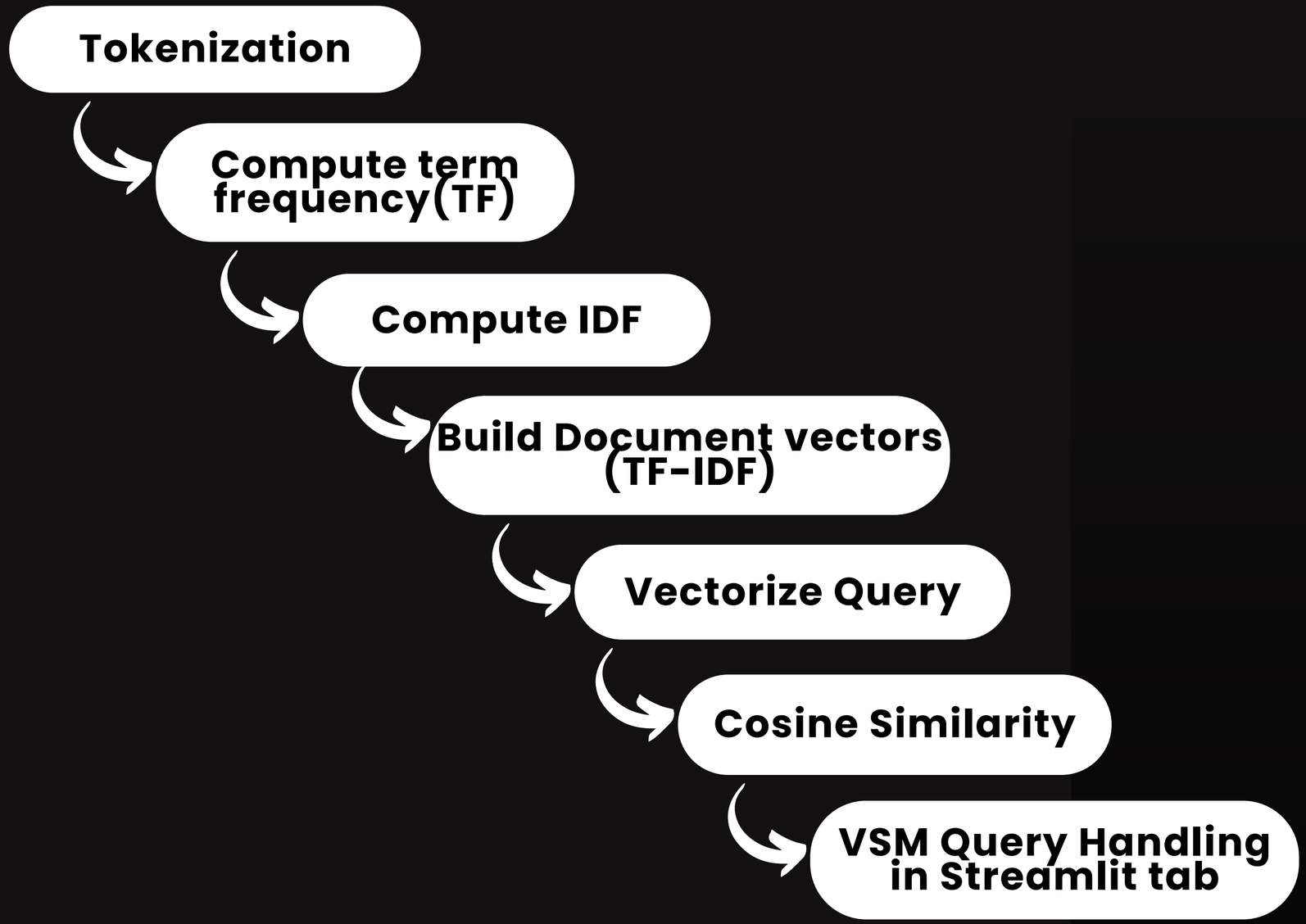
- TF (Term Frequency)- how often a term appears in a document.
- IDF(Inverse Document Frequency)- how rare the term is across all documents.
- TF-IDF – Weight of a term in a document

$$\mathrm{TF-IDF}(t,d) = \mathrm{TF}(t,d) \times \log\left(\frac{N}{\mathrm{df}(t)}\right)$$

- Cosine Similarity:

$$\cos(\theta) = \frac{\mathbf{q} \cdot \mathbf{d}}{\|\mathbf{q}\| \, \|\mathbf{d}\|}$$

VECTOR SPACE MODEL



VECTOR SPACE MODEL

```python
# Text Processing
def tokenize(text):
    return text.lower().split()

def compute_tf(doc_tokens):
    tf = Counter(doc_tokens)
    total_terms = len(doc_tokens)
    return {term: freq / total_terms for term, freq in tf.items()}

def compute_idf(inverted_index, total_docs):
    return {term: math.log(total_docs / (1 + len(set(doc_ids)))) for term, doc_ids in inverted_index.items()}

def build_document_vectors(inverted_index, metadata):
    total_docs = len(metadata)
    idf = compute_idf(inverted_index, total_docs)
    doc_vectors, doc_lengths, doc_tokens_dict = defaultdict(dict), {}, {}

    for doc_id in metadata:
        tokens = [term for term, postings in inverted_index.items() if int(doc_id) in postings]
        doc_tokens_dict[doc_id] = tokens
        tf = compute_tf(tokens)
        tfidf_vector = {term: tf[term] * idf.get(term, 0) for term in tf}
        doc_vectors[doc_id] = tfidf_vector
        doc_lengths[doc_id] = math.sqrt(sum(v ** 2 for v in tfidf_vector.values()))

    return doc_vectors, doc_lengths, idf, doc_tokens_dict
```

```python
def vectorize_query(query, idf):
    return {term: compute_tf(tokenize(query)).get(term, 0) * idf.get(term, 0) for term in tokenize(query)}

def cosine_similarity(q_vec, d_vec, d_len):
    dot_product = sum(q_vec.get(term, 0) * d_vec.get(term, 0) for term in q_vec)
    q_len = math.sqrt(sum(v ** 2 for v in q_vec.values()))
    return 0 if q_len == 0 or d_len == 0 else dot_product / (q_len * d_len)
```

QUERY LIKELIHOOD MODEL

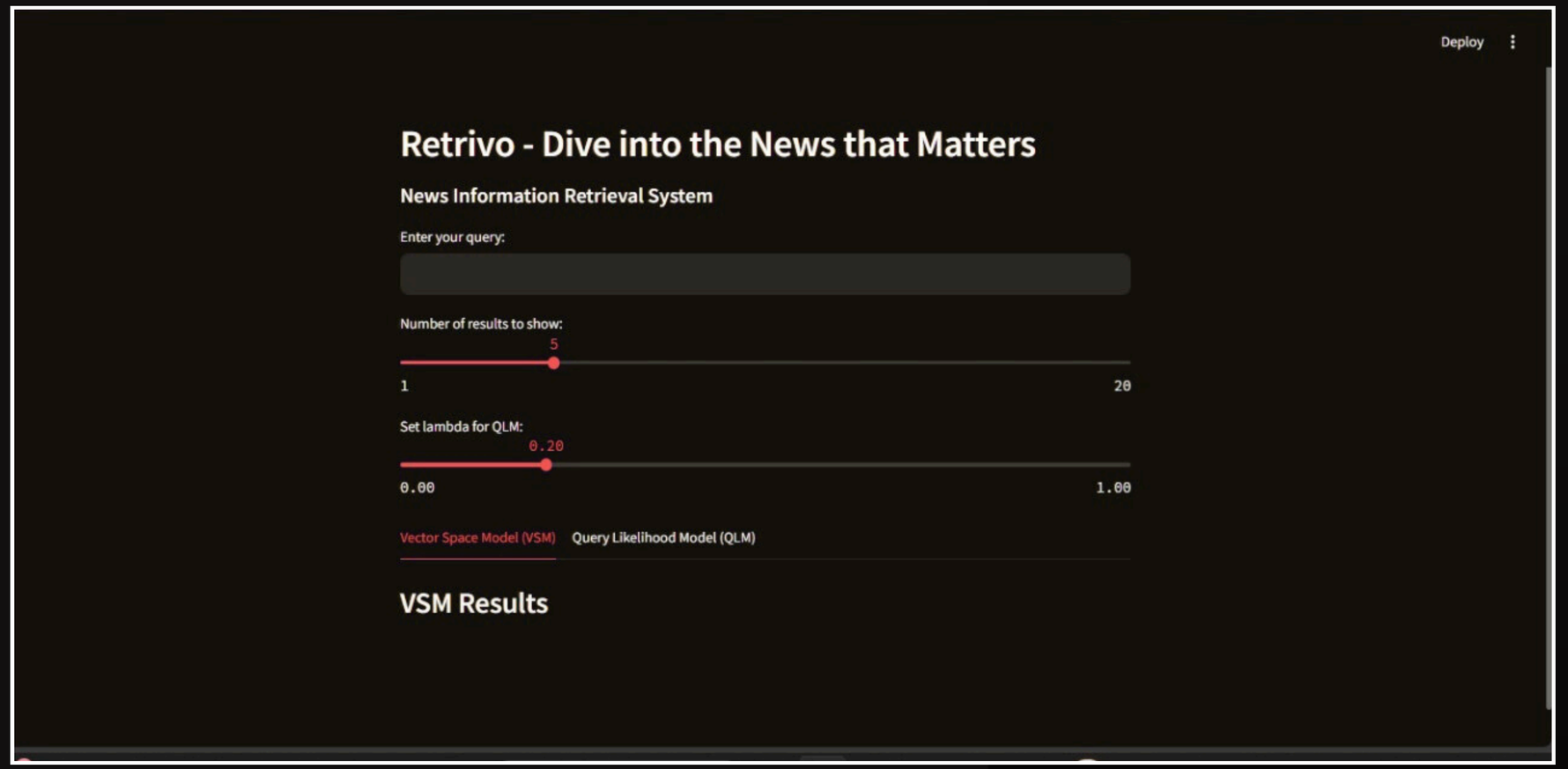
- Ranks documents based on how likely they are to generate the user's query
- Each document is treated as a language model.
- Calculates how likely a document would generate the given query

$$\log P(Q|D) = \sum_{w \in Q} \log \left[(1-\lambda) \cdot \frac{f(w,D)}{|D|} + \lambda \cdot \frac{f(w,C)}{|C|} \right]$$

QUERY LIKELIHOOD MODEL

```python
def query_likelihood(query, doc_tokens, collection_model, collection_total, lambda_param=0.2):
    doc_len, doc_model = len(doc_tokens), Counter(doc_tokens)
    return sum(
        math.log(
            (1 - lambda_param) * (doc_model.get(term, 0) / doc_len if doc_len > 0 else 0)
            + lambda_param * (collection_model.get(term, 0) / collection_total)
            )
            for term in tokenize(query)
            if collection_model.get(term, 0) > 0
    )
```

STREAMLIT APP



CHALLENGES & SOLLUTIONS

- Links displayed on the IR system were clickable, however they did not open the required html file from the local storage (file:///...)

Serving all the pages on local sever to allow the browser to access them directly.

link local - http://localhost:8080/all_pages/doc1.html

```
(venv) PS C:\Users\oketh\OneDrive\Desktop\HNDDS\Information Retreival and Analytics\CW2> cd all_pages
(venv) PS C:\Users\oketh\OneDrive\Desktop\HNDDS\Information Retreival and Analytics\CW2\all_pages> python -m http.server
8000
Serving HTTP on :: port 8000 (http://[::]:8000/) ...
```

CHALLENGES & SOLLUTIONS

- Every time the page refreshed or a user interacted Streamlit reloaded everything causing long delays and computation wastage.

Used caching to cache the result instead of re-running
@st.cache_data

```python
# Data Loading
@st.cache_data
def load_data():
    with open("merged_index.json", "r", encoding="utf-8") as f:
        inverted_index = json.load(f)["inverted_index"]

    metadata_df = pd.read_csv("documents_local.csv")
    metadata = metadata_df.set_index("DocumentID").to_dict("index")
    return inverted_index, metadata
```

```python
@st.cache_data
def build_cached_models(inverted_index, metadata):
    doc_vectors, doc_lengths, idf, doc_tokens_dict = build_document_vectors(inverted_index, metadata)
    collection_model, collection_total = build_collection_language_model(doc_tokens_dict)
    return doc_vectors, doc_lengths, idf, doc_tokens_dict, collection_model, collection_total
```

CONCLUSION

- The News Information Retrieval System helps find relevant news articles from CNN, BBC, and Al-Jazeera using two models: Vector Space Model and Query Likelihood Model. It processes user queries and ranks results based on relevance. The system uses a clean Streamlit interface, making it easy to search and compare results using both models.

THANK YOU


products: storeProducts
render() {
return (
<React.Fragment>

<Title
<div
</ProductC

</React.Fragment>
)