SearchWright Documentation

Introduction

The search engine is designed to efficiently search and retrieve relevant documents from a dataset based on user input. It supports multi-word queries, ranks results using a TF-IDF (Term Frequency-Inverse Document Frequency) algorithm, and handles stop words effectively. The system is implemented in Python. This documentation explains the working of the system and the processes involved in query handling. This system uses 190K+ medium article dataset from Kaggle.

System Components

1. Data Structures

Lexicon

- A dictionary mapping words to unique Word IDs.
- Purpose: Efficient word lookups.

Inverted Index

- A dictionary mapping Word IDs to lists of Document IDs.
- Purpose: Quickly find documents containing specific words.

Forward Index

- A dictionary mapping Document IDs to term frequency details (Word ID and weight pairs).
- The weigh is calculated using 4n+3n+2o+p where the 'n' is the frequency of word in title, 'm' is the frequency of word in authors, 'o' is the frequency of word in author and 'p' is the frequency of word in text
- Purpose: Retrieve word frequency information for ranking.

Dataset

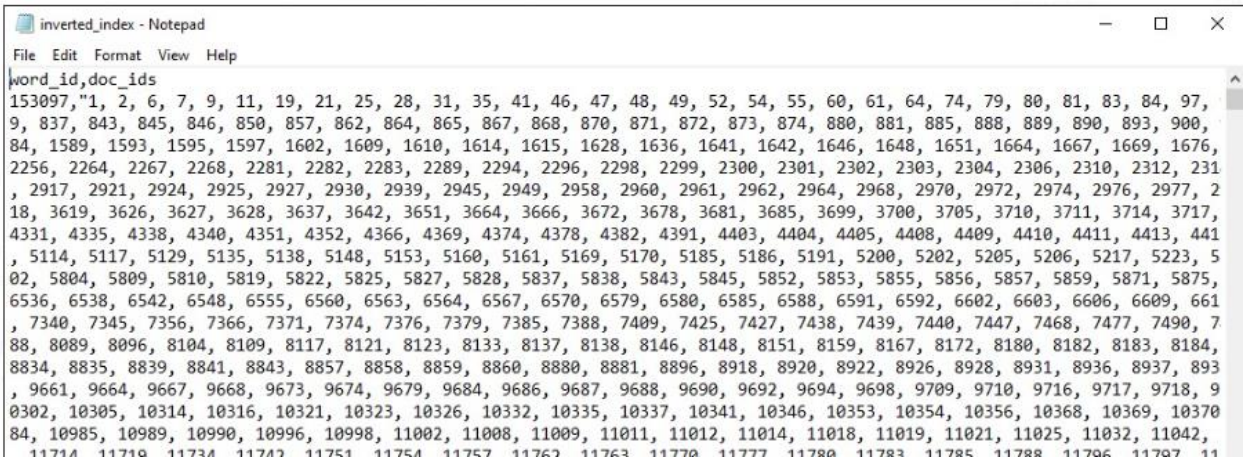
- A CSV file containing document metadata, including URLs.
- Purpose: Provide human-readable links to documents.

2. Files

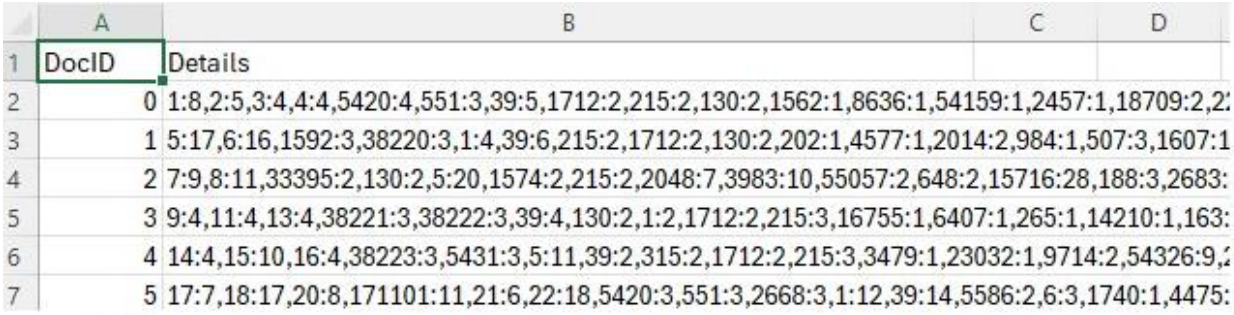
- **Lexicon.csv**: Contains words and their corresponding Word IDs.

69685	cybele	188475
69686	cybella	268546
69687	cybenko	223490
69688	cyber	3321
69689	cyberabad	121399
69690	cyberark	179098
69691	cyberattack	35315
69692	cyberattacks	105514
69693	cyberbullied	241707
69694	cyberbullies	205993
69695	cyberbullying	90772
69696	cyberbyte	268552
69697	cybercash	107078
69698	cyberciti	222508
69699	cybercodetwin	195428
69700	cybercrime	33957

- **InvertedIndex.csv**: Contains Word IDs and lists of associated Document IDs.



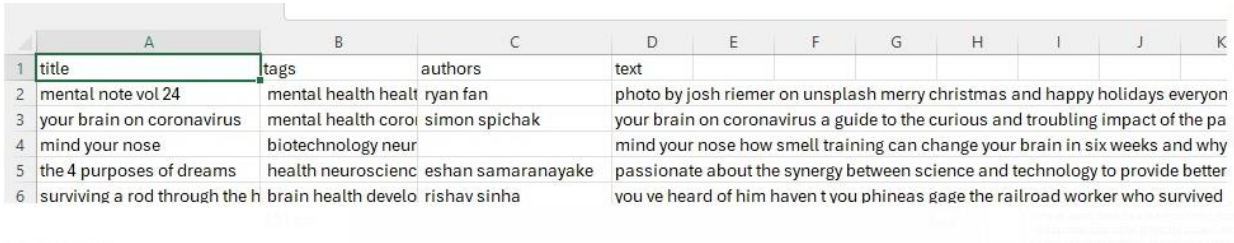
- **ForwardIndex.csv**: Contains Document IDs and term frequency details.



	A	B	C	D
1	DocID	Details		
2	0	1:8,2:5,3:4,4:4,5420:4,551:3,39:5,1712:2,215:2,130:2,1562:1,8636:1,54159:1,2457:1,18709:2,2		
3	1	5:17,6:16,1592:3,38220:3,1:4,39:6,215:2,1712:2,130:2,202:1,4577:1,2014:2,984:1,507:3,1607:1		
4	2	7:9,8:11,33395:2,130:2,5:20,1574:2,215:2,2048:7,3983:10,55057:2,648:2,15716:28,188:3,2683:		
5	3	9:4,11:4,13:4,38221:3,38222:3,39:4,130:2,1:2,1712:2,215:3,16755:1,6407:1,265:1,14210:1,163:		
6	4	14:4,15:10,16:4,38223:3,5431:3,5:11,39:2,315:2,1712:2,215:3,3479:1,23032:1,9714:2,54326:9,2		
7	5	17:7,18:17,20:8,171101:11,21:6,22:18,5420:3,551:3,2668:3,1:12,39:14,5586:2,6:3,1740:1,4475:		

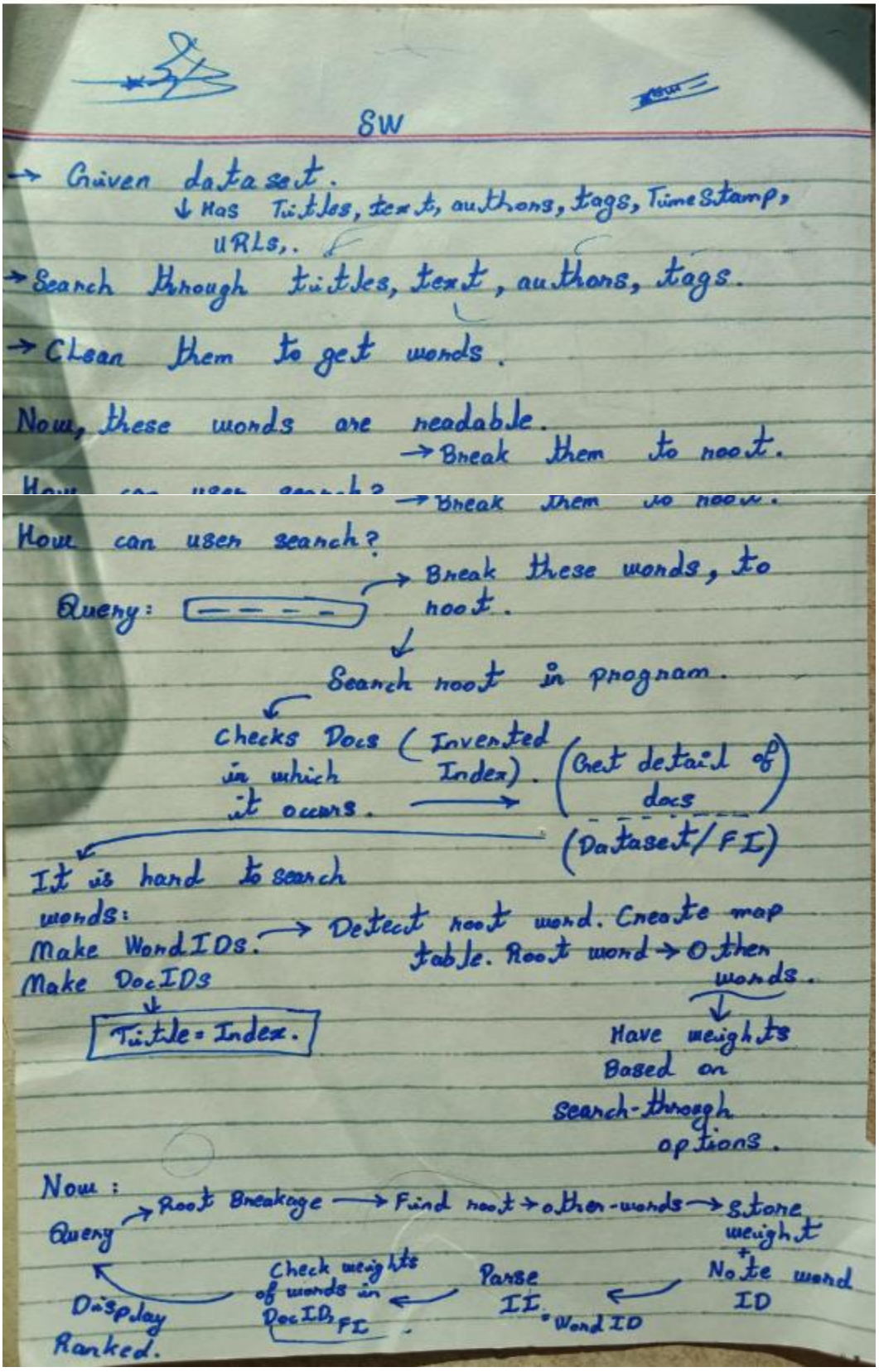
- **CleanedSubDataset.csv**: Contains document metadata, including URLs. Removed punctuation and null values.

- **ExtractedCleaneedColumns.csv** This has only four columns and is the subset of CleanedSubDataset.csv file but has fetch those columns only on the basis of which ranking is implemented.



	A	B	C	D	E	F	G	H	I	J	K
1	title	tags	authors	text							
2	mental note vol 24	mental health healt	ryan fan	photo by josh riemer on unsplash merry christmas and happy holidays everyon							
3	your brain on coronavirus	mental health coro	simon spichak	your brain on coronavirus a guide to the curious and troubling impact of the pa							
4	mind your nose	biotechnology neur		mind your nose how smell training can change your brain in six weeks and why							
5	the 4 purposes of dreams	health neuroscienc	eshan samaranayake	passionate about the synergy between science and technology to provide better							
6	surviving a rod through the h	brain health develo	rishav sinha	you ve heard of him haven t you phineas gage the railroad worker who survived							

System Design



Query Processing Workflow

1. Initialization

The system loads the lexicon, inverted index, and forward index into memory. Each file is read line-by-line to populate the respective data structures.

2. Handling User Input

- **Stop Word Removal**: User input is tokenized into individual words, and common stop words such as "a," "the," and "is" are filtered out.
- **Normalization**: Remaining words are converted to lowercase for consistent processing.

3. Processing the Query

For each word in the query:

1. **Lexicon Lookup**:
    - Check if the word exists in the lexicon.
    - If not found, the word is ignored, and the user is notified.
2. **Inverted Index Retrieval**:
    - Retrieve Document IDs associated with the Word ID from the inverted index.
3. **Forward Index Retrieval**:
    - Extract term frequency details for each Document ID from the forward index.
4. **TF-IDF Calculation**:
    - Compute the TF-IDF score for each Document ID:
        - TF: Term Frequency (frequency of the word in the document).
        - IDF: Inverse Document Frequency ("logarithmic measure of the word's rarity across all documents").
    - Combine scores for multi-word queries by summing TF-IDF values across words.

4. Ranking Documents

Documents are ranked in descending order of their cumulative TF-IDF scores. The top 30 results are selected for display.

5. Fetching Document Metadata

Using the ranked Document IDs:

- Adjust Document IDs to match the dataset's row numbering (0-based indexing).
- Retrieve corresponding URLs from the dataset.

6. Displaying Results

- Results include the document URL and its TF-IDF score.
- After displaying results, the program prompts the user to search for a new search query. Users can type “exit” to terminate the program.

Example Workflow

Input:

User types: "mental health in the workplace"

Processing:

1. **Stop Word Removal**:
    - Filtered query: "mental health workplace"
2. **Lexicon Lookup**:
    - Words found: "mental," "health," "workplace"
3. **Inverted Index Retrieval**:
    - Document IDs fetched for each word.
4. **TF-IDF Calculation**:
    - Compute scores for documents containing these words.
5. **Ranking**:
    - Documents ranked by cumulative TF-IDF scores.
6. **Fetching URLs**:
    - Top 5 document links retrieved.

Output:

- **Document 1**: Link: https://example.com/doc1, Score: 3.4567
- **Document 2**: Link: https://example.com/doc2, Score: 2.9876
- **Document 3**: Link: https://example.com/doc3, Score: 2.1234

Key Features

1. **Multi-word Query Support**:
    - Combines results for multiple words and ranks documents based on their relevance.
2. **Stop Word Removal**:
    - Filters out common words that do not contribute to meaningful search results.
3. **TF-IDF Ranking**:
    - Ensures relevant documents are prioritized by considering term frequency and rarity.

Conclusion

This Query Processing System effectively handles large datasets and multi-word queries, ranking results using a robust TF-IDF algorithm. The system's ability to filter stop words, process queries repeatedly, and rank results ensures a user-friendly and efficient search experience.

For further customization or troubleshooting, refer to the provided source code. Feel free to enhance the system with additional features such as phrase matching or fuzzy search.