Enhanced Document Ranking System

Presented By:
Shahzaib Irfan 2021-CS-7

Agenda

1. Why do we need this system
2. Introduction
3. Objectives
4. Scope
5. Assignment Modules
6. Data Flow Diagram
7. Conclusion

Why do we need this system

- Searching a large library of text documents manually is time consuming.
- We need an algorithm to find relevant documents efficiently.
- TF-IDF helps rank documents by their relevance to a query.

InetDaemon.Com - Tutorials: Internet, Telecom, Satellite, Computers, Troubleshooting - Mozilla Firefox
InetDaemon.Com

Introduction

A document search engine enables efficient information retrieval by allowing users to search for keywords or phrases. The core of a search engine is ranking documents by their relevance to user query.

Objectives

- Goal:
  - Develop a basic search engine to rank documents according to their relevance to user query.
  - Optimize user experience by providing relevant and accurate search results.
- Important Tasks:
  - Implement scoring to setup rankings in documents.

Scope

- Scope:
  - Build a simple, web based search engine.
  - Provide keyword matching and TF-IDF scoring.

Assignment modules

- Text Preprocessing
- Gather Data
- TF - IDF Implementation
- Searching
- Ranking Documents

Pre-Processing

Preprocessing is a crucial first step in building a search engine or any system dealing with large text data. It involves cleaning and organizing text to make it more "search-friendly."

Pre-Processing Techniques:

- Tokenization
- Stop Words Removal

Pre-Processing (Continued…)

Tokenization:

Split text into individual words or "tokens" that can be indexed.

Lets understand with an example:

“Ali plays video games in the evening” ⇨ [“Ali”, “plays”, “video”, “games”, “in”, “the”, “evening”]

Pre-Processing (Continued…)

Stop Words Removal:

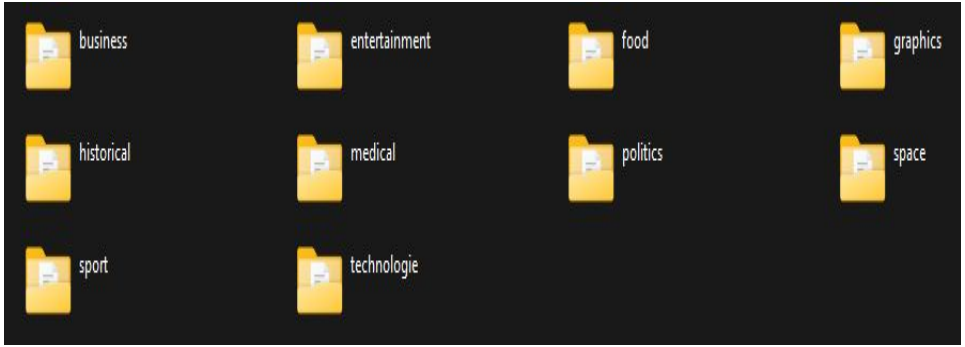
Removes words that do not contribute much meaning in text analysis.

Lets understand further with previous example:

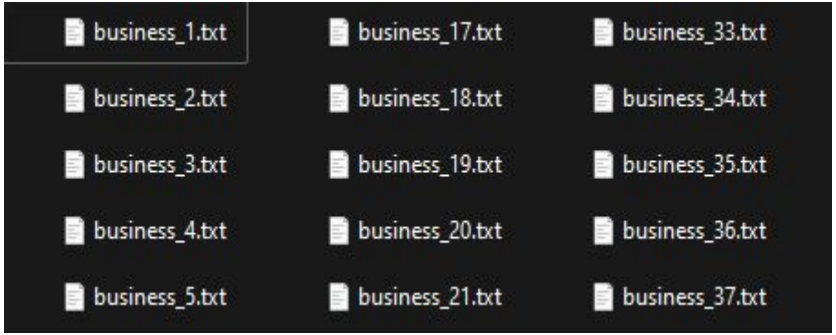
[“Ali”, “plays”, “video”, “games”, “in”, “the” , “evening”] ⇨
[“Ali”, “plays”, “video”, “games”, “evening”]

Gather Data

- Text files stored in a directory, containing different folders each representing a category.
- Each folder contains 100 text documents.



business
entertainment
food
graphics
historical
medical
politics
space
sport
technologie



business_1.txt
business_2.txt
business_3.txt
business_4.txt
business_5.txt
business_17.txt
business_18.txt
business_19.txt
business_20.txt
business_21.txt
business_33.txt
business_34.txt
business_35.txt
business_36.txt
business_37.txt

TF-IDF Implementation

- TF (Term Frequency):
    - Measures how often a term appears in a document relative to total words.
    - Formula:
        - Term = count of Term / total words in document.

TF-IDF Implementation (Continued…)

- TF(Example):
  - Suppose you have X number of coins, and you want to rank similar coins in ascending orders.
  - Coin Y appears Z times in the coin set, so its term frequency would be:
    - Coins[Y] = Z (count of Y coins) / X (total coins)

TF-IDF Implementation (Continued…)

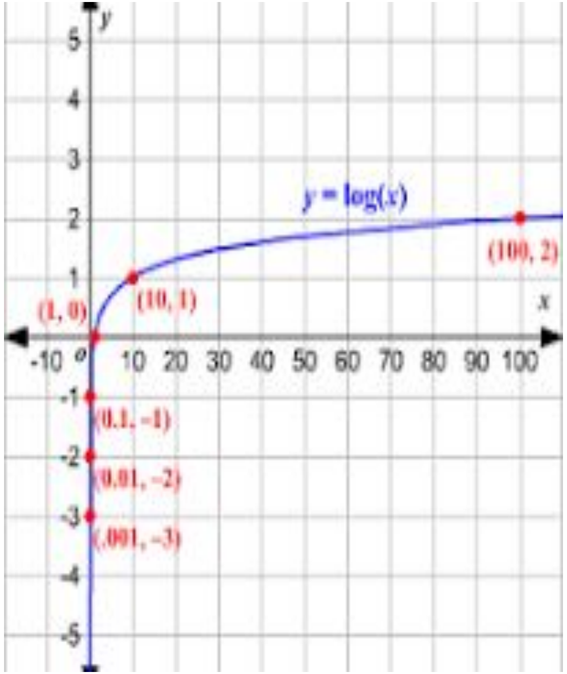
- IDF (Inverse Document Frequency):
  - Measures term importance across documents by giving higher scores to terms with fewer appearance in dataset.
  - Formula:
    - Term = log(Total Documents / (1 + Documents Containing Term)).

TF-IDF Implementation (Continued…)

- IDF (Formula Explanation):
    - **Total Documents**: Total documents in corpus.
    - **Documents Containing Term**: Simply it is count of terms in the complete corpus.
    - **Adding 1**: To avoid division by 0.
    - **Logarithmic Function**: To penalize common terms and reward rare terms.

TF-IDF Implementation (Continued…)

- IDF (Logarithm Function):
  - It penalizes higher values more and lower values less.
  - Example:
    - log(100) = 2
    - log(10) = 1



TF-IDF Implementation (Continued…)

- IDF (Calculated Example):
    - Consider two words “machine” and “data”.
    - Documents with “machine” = 9.
    - Documents with “data” = 499.
    - IDF[machine] = log(1000 /(1 + 9)) => log(100) => 2.
    - IDF[data] = log(1000 / (1 + 499)) => log(2) => 0.3.

So, word “machine” is more relevant to be retrieved because of its higher relevancy score.

TF-IDF Implementation (Continued…)

- TF-IDF (Term Frequency - Inverse Document Frequency):
    - TF-IDF = TF * IDF

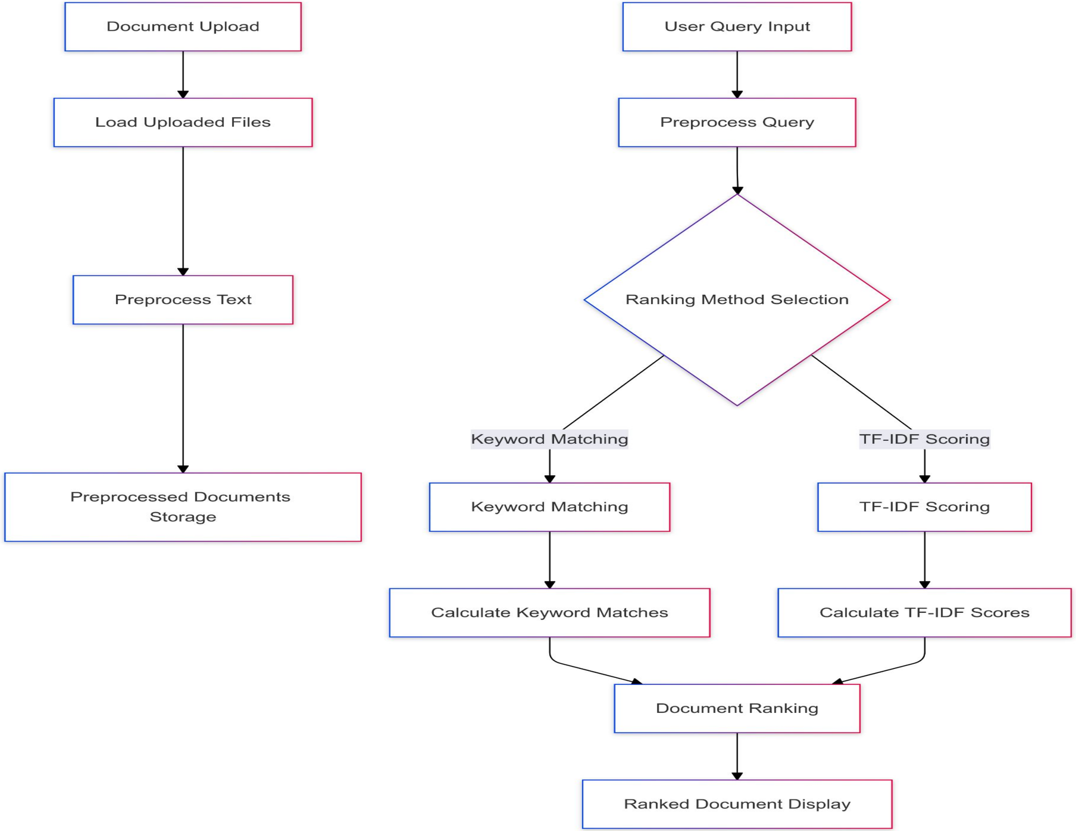
How Searching is Performed

- Content-Based Search:
  - User provides a query containing one or more keywords.
  - Keywords then scanned on documents and matched results are retrieved.

Ranking

- Keyword Matching
    - Documents are ranked on the basis of count of keywords matched in corpus.
- TF - IDF Ranking
    - TF - IDF calculated for each keyword and total score is added to rank documents with higher TF - IDF score.

Data Flow Diagram



Summary & Conclusion

- Summary:
  - Basic document search engine implemented with keyword matching.
  - TF-IDF used for relevance scoring to improve search accuracy.
- Content-Based Search:
  - This assignment demonstrates foundational IR concepts with indexing and search.
  - Offers navigation for adding and retrieving documents.