Indexing

Presented By:
Shahzaib Irfan 2021-CS-7

Indexing

- Lets say we want to search something from a collection of text documents or from a database where text documents are stored.

- What would be our approach?
  - Start from document 1.
  - Read line by line.
  - Stop if desired content is found.

Indexing

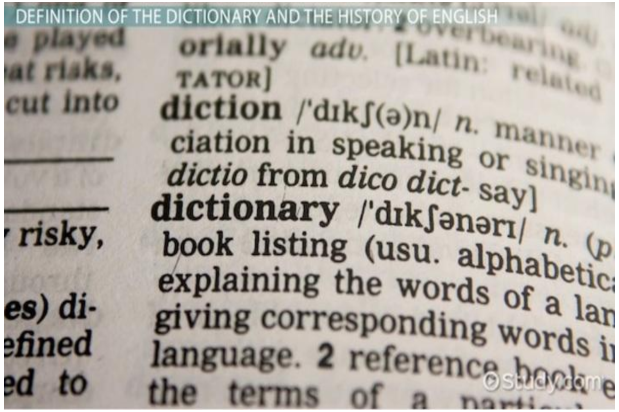
- Problem with this approach?

- Reading each and every single document in a collection of documents is time and resource consuming.
- That is not how every search engine is implemented.
- Take into consideration an example of google search engine, it does not search in its database from start to end.

Indexing

- So what is the solution for this problem?

- Building and using an Index
- Take an example of dictionary.
- Each and every word is listed in alphabetical order, page numbers are mentioned against a word.
- When you visit that page in dictionary, you would see it has different word forms such as alter, altered etc.



DEFINITION OF THE DICTIONARY AND THE HISTORY OF ENGLISH

e played
at risks,
cut into

risky,

es) di-
efined
ed to

orially adv. [Latin: related
TATOR]
diction /ˈdɪkʃ(ə)n/ n. manner
ciation in speaking or singin
dictio from dico dict- say]
dictionary /ˈdɪkʃənərɪ/ n. (p
book listing (usu. alphabetic
explaining the words of a lan
giving corresponding words i
language. 2 reference book
the terms of a partic

©Study.com

Indexing

Formal Definition of Indexing:

“A data structure technique that is used for quickly retrieving entries from database files using some attributes that have been indexed”.

Assignment 1

Information Retreival
Suprevised By: Dr. Syed Khaldoon Khurshid

Introduction

- Build a document search engine.
- Use indexing to implement a simple search engine.

Objectives

- Goal:
  - Develop a basic search engine to search documents by title and content.
- Important Tasks:
  - Implement Indexing for fast lookup.
  - Implement scoring to setup important terms for indexing.

Scope

- Scope:
    - Build a simple, web based search engine.
    - Provide title and content based search.

Assignment modules

- Text Preprocessing
- Expansion with Synonyms
- Gather Data
- Term Frequency - Inverse Document Frequency Implementation
- Indexing
- Searching

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
- Lemmatization

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

Pre-Processing (Continued…)

Lemmatization:

Reduce words to their root form.

Lets understand with few examples:

- “Running” ⇨ “Run”
- “Children” ⇨ “Child”
- “Better” ⇨ “Good”

Expansion with Synonyms

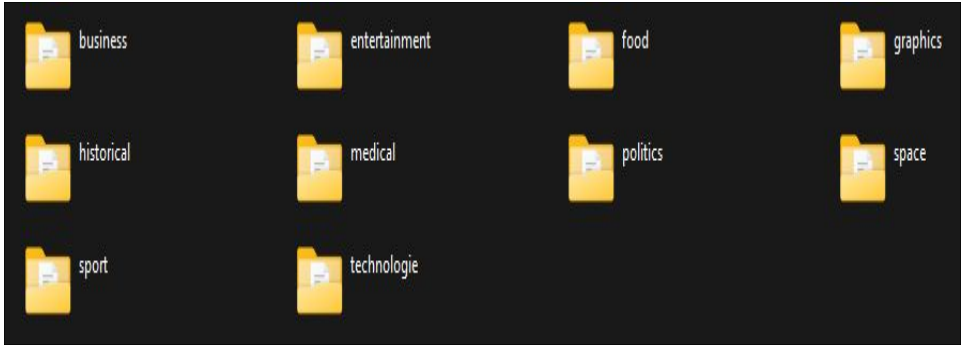
Query expansion with synonyms helps in better searching in documents even if exact matches are not found.

Practical Scenario:

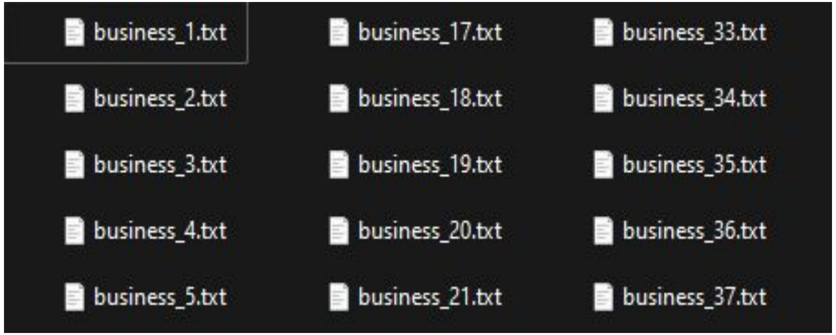
1. **User Query**: “treatment”.
2. **Expanded Query**: “treatment, therapy, medication, cure”
3. **Result**: Documents with “treatment”, “therapy”, “medication”, “cure” will also be retrieved by search engine.

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

Synonyms & Query Expansion

- Retrieves synonyms for a given word from WordNet.
- Expands search queries to increase matches, even if the exact words is not used in the document.

- Example:
  - Search for “hope” includes results for synonyms like “aspiration” and “wish” etc.

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…)

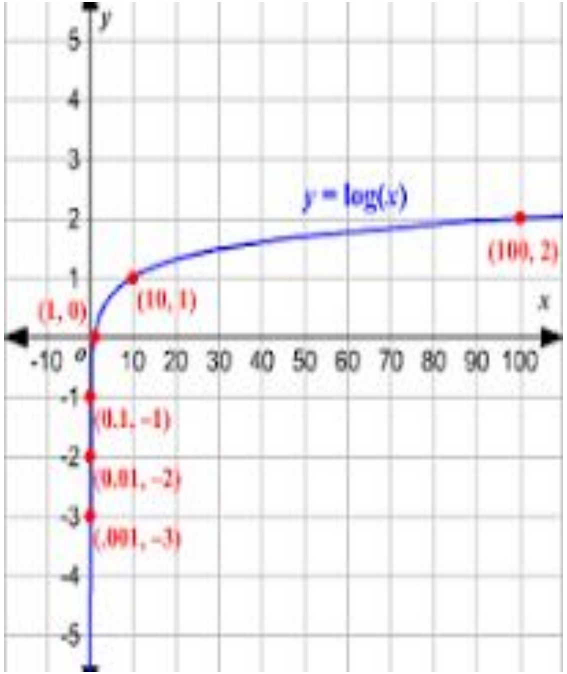
- IDF (Inverse Document Frequency):
    - Measures term importance across documents by giving higher scores to terms with fewer appearance in dataset.
    - Formula:
        - Term = log(Total Documents / (1 + Documents Containing Term)).

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…)

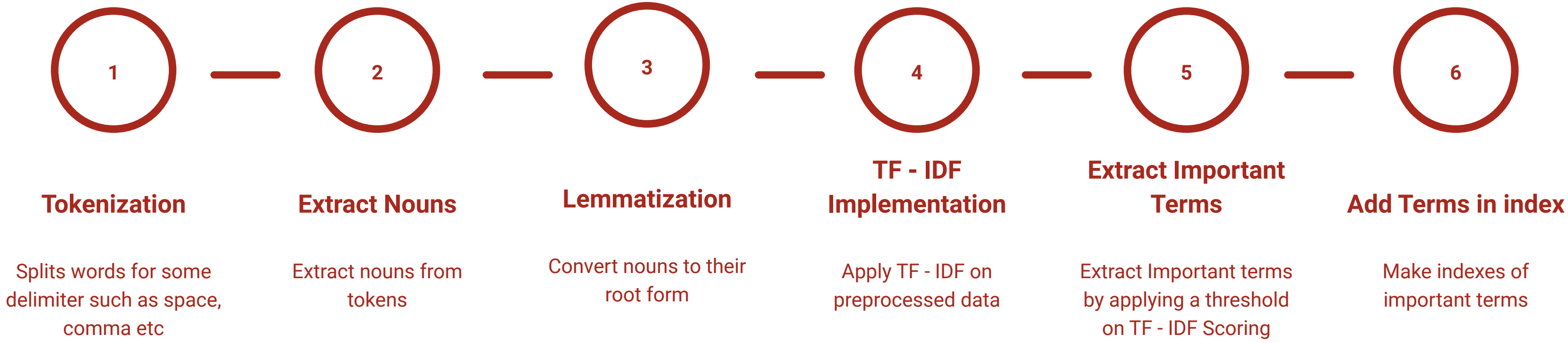
- IDF (Calculated Example):
    - Consider two words “machine” and “data”.
    - Documents with “machine” = 9.
    - Documents with “data” = 499.
    - IDF[machine] = log(1000 /(1 + 9)) => log(100) => 2.
    - IDF[data] = log(1000 / (1 + 499)) => log(2) => 0.3.

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

Indexing



1

Tokenization

Splits words for some delimiter such as space, comma etc

2

Extract Nouns

Extract nouns from tokens

3

Lemmatization

Convert nouns to their root form

4

TF - IDF Implementation

Apply TF - IDF on preprocessed data

5

Extract Important Terms

Extract Important terms by applying a threshold on TF - IDF Scoring

6

Add Terms in index

Make indexes of important terms

How Searching is Performed

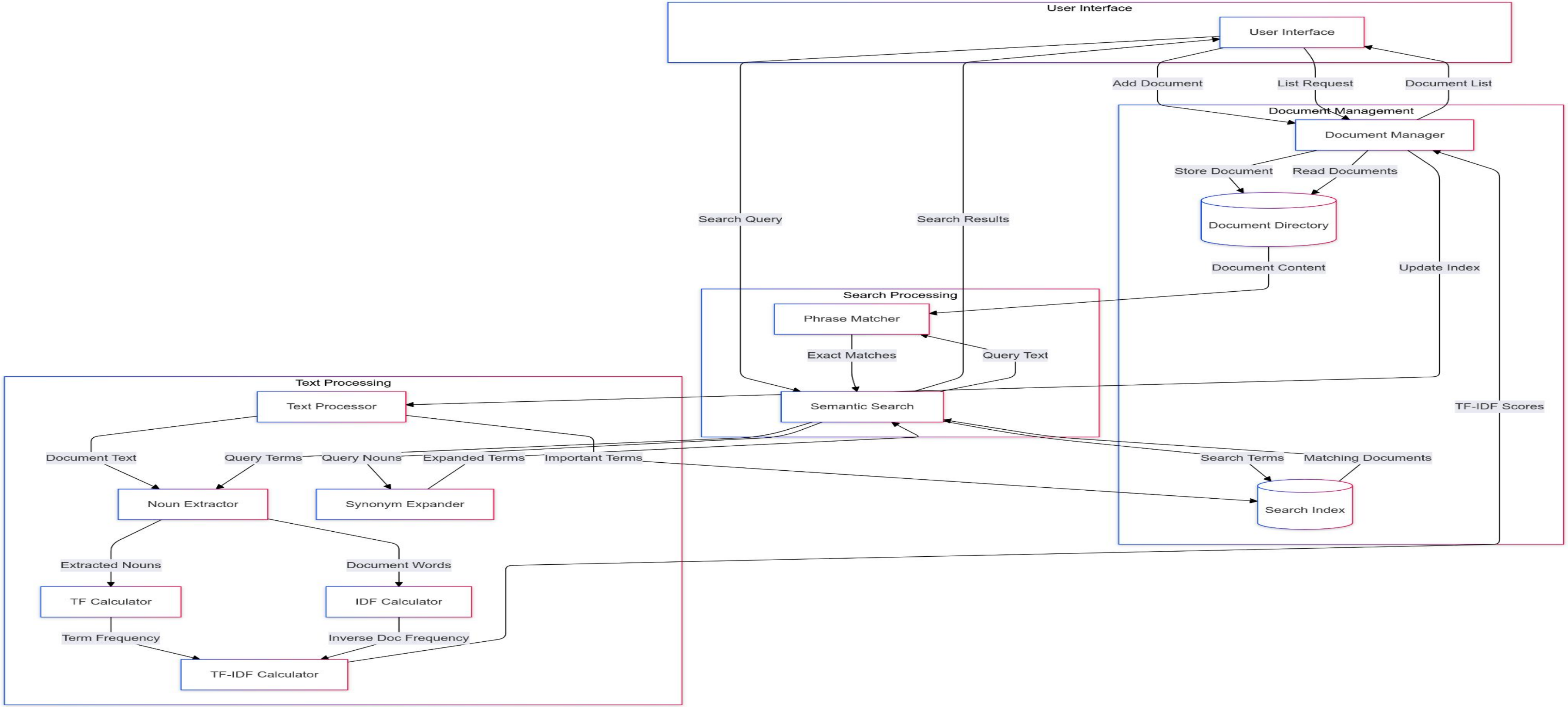
- Title-Based Search:
    - Checks if title contains the query term.
    - Returns exact matches for document titles.

- Content-Based Search:
    - Matches phrases directly or uses expanded terms with synonyms.
    - Looks up each term in the index to retrieve documents with matching terms.

Summary & Conclusion

- Summary:
  - Basic document search engine implemented with indexing and synonym expansion.
  - 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.

Data Flow Diagram



Thank You

Any Questions?