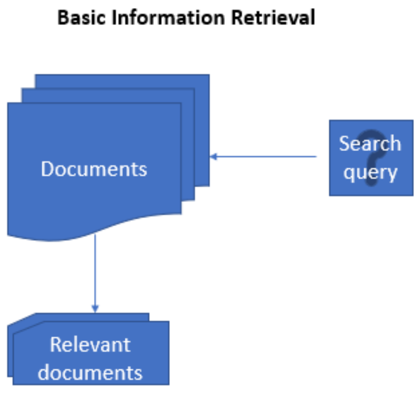
Dept of computer Science Engineering
Hong Kong University of Science and Technology

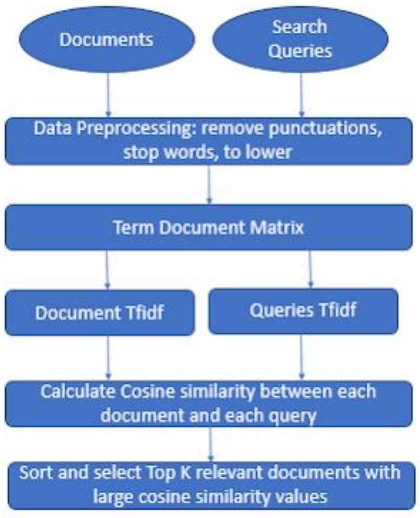
Parthiban Supriya
20567171

CSIT 6000I Search Engines and Applications

Homework 2 Fall Semester 2018

The objective of this homework is to create a basic search engine or document retrieval system using Vector space model. We have a text collection (collection-100.txt) with 100 documents and a query file (query-10.txt) with 5 queries. We need to build an inverted index and retrieve the top 3 relevant documents for each search query using cosine similarity. For each of these documents we must display the document ID, five highest weighted keywords of the document and the posting lists, the number of unique keywords in the document, the magnitude (L2 norm) of the document vector, and the similarity score.




Documents
Search Queries
Data Preprocessing: remove punctuations, stop words, to lower
Term Document Matrix
Document Tfidf
Queries Tfidf
Calculate Cosine similarity between each document and each query
Sort and select Top K relevant documents with large cosine similarity values

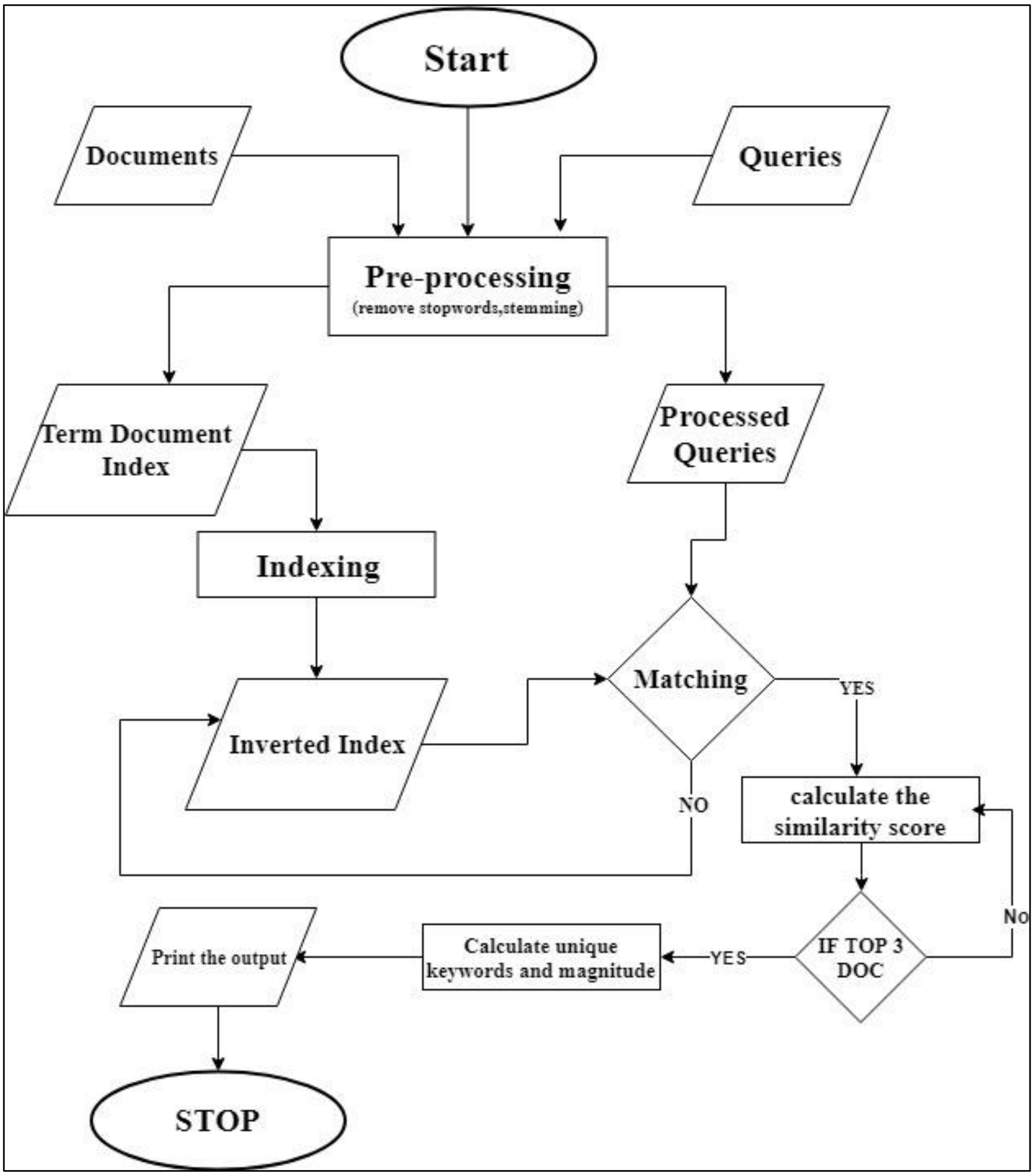
System Design

Design: The first step is to fetch all documents and queries. We then pre-process these data by removing the punctuations. The document is split into words. Words with length lesser than 4 from the documents (e.g. : a, am, an, also, any, and etc..). These words are called Stop words and have little value in search, so we strip them. Then we remove the ending 's' of the words. This process is called stemming (although we don't implement here) A stemmer takes words and tries to reduce them to their base or root. Words which have a common stem often have similar meanings. Example: connected, connecting and connections. The pre-processed words are then stored in a dictionary with docID as the key and the keywords & their positions as the value also called the term document index. The pre-processing is repeated for queries and a list of query terms is created.

The next step is the creation of the inverted index. We use the term document index dictionary to iterate through every keyword. If the keyword is not present a new key value pair is added to the inverted index dictionary. If the word already exists, the new document ID and the position is appended. And if the word and the document are already present then the new position is appended. We then calculate the tfmax of all the document. We also calculate the tf, idf, weight, number of unique keywords and magnitude of each of the document and store them.

Now we compare the words in the query with the words in the inverted index and find the documents containing the query. We then find the tf, idf and similarity score of all the query and document pair. Based on the cosine similarity the top 3 documents are found. Once we have got the ID of the documents, we find the top five keywords in the document by using their weights. All the required outputs like magnitude, similarity score are then printed.

Flowchart :



(c) Output:

```
In [3]: runfile('D:/Hong Kong/HKUST/CSIT 6000I Search Engines/
20567171_hw2/20567171_hw2_pythoncode.py', wdir='D:/Hong Kong/HKUST/CSIT 6000I Search
Engines/20567171_hw2')
****************************************************************************************
query: ['bank']
DID: Doc84
top 5 keywords
billion-> [('Doc29', [6]), ('Doc36', [0, 1]), ('Doc59', [1]), ('Doc64', [2, 3]),
('Doc84', [3, 4, 4]), ('Doc90', [4]), ('Doc91', [5]), ('Doc98', [6])]
bank-> [('Doc20', [33]), ('Doc30', [0, 1]), ('Doc53', [1]), ('Doc60', [2, 3]), ('Doc73',
[3]), ('Doc84', [4, 5, 5]), ('Doc89', [5]), ('Doc90', [6]), ('Doc92', [7])]
venezuela-> [('Doc84', [0]), ('Doc88', [0]), ('Doc89', [1])]
advisory-> [('Doc84', [4]), ('Doc90', [0])]
agreed-> [('Doc84', [7])]
Unique keywords in the document: 24
Magnitude of the document vector: 10.200
Similarity score: 0.341
..........................................................................................
DID: Doc60
top 5 keywords
bank-> [('Doc20', [33]), ('Doc30', [0, 1]), ('Doc53', [1]), ('Doc60', [2, 3]), ('Doc73',
[3]), ('Doc84', [4, 5, 5]), ('Doc89', [5]), ('Doc90', [6]), ('Doc92', [7])]
debt-> [('Doc19', [26]), ('Doc28', [0]), ('Doc29', [1]), ('Doc30', [2, 3, 3]), ('Doc36',
[3]), ('Doc37', [4]), ('Doc52', [5]), ('Doc58', [6]), ('Doc59', [7]), ('Doc60', [8, 9, 9]),
('Doc90', [9])]
even-> [('Doc30', [22]), ('Doc60', [0])]
fail-> [('Doc30', [28]), ('Doc60', [0])]
expect-> [('Doc30', [37]), ('Doc60', [0]), ('Doc63', [1]), ('Doc68', [2])]
Unique keywords in the document: 25
Magnitude of the document vector: 7.221
Similarity score: 0.321
..........................................................................................
DID: Doc30
top 5 keywords
losse-> [('Doc30', [6, 0]), ('Doc59', [0]), ('Doc60', [1])]
debt-> [('Doc19', [26]), ('Doc28', [0]), ('Doc29', [1]), ('Doc30', [2, 3, 3]), ('Doc36',
[3]), ('Doc37', [4]), ('Doc52', [5]), ('Doc58', [6]), ('Doc59', [7]), ('Doc60', [8, 9, 9]),
('Doc90', [9])]
bank-> [('Doc20', [33]), ('Doc30', [0, 1]), ('Doc53', [1]), ('Doc60', [2, 3]), ('Doc73',
[3]), ('Doc84', [4, 5, 5]), ('Doc89', [5]), ('Doc90', [6]), ('Doc92', [7])]
however-> [('Doc30', [2]), ('Doc59', [0])]
potential-> [('Doc30', [5]), ('Doc59', [0]), ('Doc95', [1])]
Unique keywords in the document: 33
Magnitude of the document vector: 8.834
Similarity score: 0.262
..........................................................................................
****************************************************************************************
query: ['stock', 'banking']
1
DID: Doc50
top 5 keywords
quickly-> [('Doc17', [8]), ('Doc50', [0])]
poor-> [('Doc17', [26]), ('Doc50', [0])]
performance-> [('Doc17', [27]), ('Doc50', [0]), ('Doc66', [1])]
pressure-> [('Doc17', [5]), ('Doc20', [0]), ('Doc50', [1]), ('Doc53', [2])]
```

```
proposed-> [('Doc17', [11]), ('Doc34', [0]), ('Doc50', [1])]
Unique keywords in the document: 19
Magnitude of the document vector: 15.238
Similarity score: 0.284
..........................................................................................
DID: Doc17
top 5 keywords
bankamerica-> [('Doc17', [0]), ('Doc18', [0]), ('Doc19', [1]), ('Doc20', [2]), ('Doc21',
[3]), ('Doc22', [4]), ('Doc23', [5, 6]), ('Doc25', [6]), ('Doc26', [7]), ('Doc27', [8]),
('Doc29', [9]), ('Doc30', [10]), ('Doc50', [11]), ('Doc51', [12]), ('Doc52', [13]),
('Doc53', [14]), ('Doc54', [15, 16]), ('Doc55', [16, 17]), ('Doc56', [17]), ('Doc57', [18,
19]), ('Doc59', [19]), ('Doc60', [20])]
corp-> [('Doc17', [1]), ('Doc36', [0]), ('Doc49', [1]), ('Doc50', [2]), ('Doc69', [3]),
('Doc75', [4, 5]), ('Doc77', [5])]
pressure-> [('Doc17', [5]), ('Doc20', [0]), ('Doc50', [1]), ('Doc53', [2])]
quickly-> [('Doc17', [8]), ('Doc50', [0])]
proposed-> [('Doc17', [11]), ('Doc34', [0]), ('Doc50', [1])]
Unique keywords in the document: 19
Magnitude of the document vector: 16.430
Similarity score: 0.263
..........................................................................................
DID: Doc19
top 5 keywords
thi-> [('Doc4', [8]), ('Doc19', [0, 1, 1]), ('Doc46', [1]), ('Doc52', [2, 3, 3]),
('Doc86', [3])]
new-> [('Doc19', [12, 0]), ('Doc52', [0, 1])]
stock-> [('Doc17', [24]), ('Doc19', [0, 1]), ('Doc20', [1]), ('Doc26', [2]), ('Doc27', [3,
4]), ('Doc36', [4]), ('Doc38', [5, 6]), ('Doc39', [6]), ('Doc40', [7]), ('Doc41', [8]),
('Doc45', [9]), ('Doc50', [10]), ('Doc52', [11, 12]), ('Doc53', [12]), ('Doc57', [13, 14,
14]), ('Doc78', [14])]
week-> [('Doc1', [4]), ('Doc2', [0]), ('Doc4', [1]), ('Doc19', [2, 3]), ('Doc28', [3]),
('Doc52', [4, 5]), ('Doc58', [5])]
fell-> [('Doc19', [2]), ('Doc52', [0])]
Unique keywords in the document: 28
Magnitude of the document vector: 8.089
Similarity score: 0.255
..........................................................................................
*******************************************************************************************
query: ['the', 'company', 'share']
DID: Doc65
top 5 keywords
share-> [('Doc38', [15, 0]), ('Doc39', [0, 1]), ('Doc40', [1, 2]), ('Doc41', [2]), ('Doc44',
2
[3]), ('Doc63', [4]), ('Doc65', [5, 6]), ('Doc66', [6]), ('Doc79', [7, 8]), ('Doc80', [8]),
('Doc81', [9, 10, 10]), ('Doc95', [10])]
repeated-> [('Doc65', [1])]
projection-> [('Doc65', [5])]
thirdquarter-> [('Doc65', [7])]
probably-> [('Doc65', [10])]
Unique keywords in the document: 17
Magnitude of the document vector: 10.033
Similarity score: 0.258
..........................................................................................
DID: Doc40
top 5 keywords
computer-> [('Doc39', [0]), ('Doc40', [0, 1]), ('Doc42', [1]), ('Doc43', [2, 3])]
terminal-> [('Doc39', [1]), ('Doc40', [0, 1]), ('Doc42', [1]), ('Doc43', [2, 3])]
```

```
company-> [('Doc37', [1]), ('Doc38', [0, 1]), ('Doc40', [1, 2]), ('Doc41', [2]),
('Doc43', [3]), ('Doc45', [4]), ('Doc49', [5, 6]), ('Doc61', [6]), ('Doc65', [7]), ('Doc68',
[8, 9, 9, 9]), ('Doc71', [9]), ('Doc76', [10]), ('Doc78', [11]), ('Doc79', [12, 13])]
share-> [('Doc38', [15, 0]), ('Doc39', [0, 1]), ('Doc40', [1, 2]), ('Doc41', [2]), ('Doc44',
[3]), ('Doc63', [4]), ('Doc65', [5, 6]), ('Doc66', [6]), ('Doc79', [7, 8]), ('Doc80', [8]),
('Doc81', [9, 10, 10]), ('Doc95', [10])]
exercisable-> [('Doc40', [6]), ('Doc41', [0])]
Unique keywords in the document: 28
Magnitude of the document vector: 13.304
Similarity score: 0.256
..........................................................................................
DID: Doc81
top 5 keywords
share-> [('Doc38', [15, 0]), ('Doc39', [0, 1]), ('Doc40', [1, 2]), ('Doc41', [2]), ('Doc44',
[3]), ('Doc63', [4]), ('Doc65', [5, 6]), ('Doc66', [6]), ('Doc79', [7, 8]), ('Doc80', [8]),
('Doc81', [9, 10, 10]), ('Doc95', [10])]
dlr-> [('Doc5', [21]), ('Doc6', [0, 1, 1, 1]), ('Doc7', [1, 2, 2]), ('Doc10', [2, 3, 3]),
('Doc11', [3, 4, 4]), ('Doc29', [4, 5, 5]), ('Doc36', [5, 6]), ('Doc39', [6]), ('Doc40',
[7]), ('Doc41', [8]), ('Doc42', [9]), ('Doc44', [10]), ('Doc46', [11]), ('Doc59', [12, 13,
13]), ('Doc64', [13, 14]), ('Doc66', [14]), ('Doc68', [15]), ('Doc71', [16]), ('Doc80', [17,
18]), ('Doc81', [18, 19, 19, 19]), ('Doc84', [19, 20]), ('Doc98', [20])]
nine-> [('Doc81', [3])]
declined-> [('Doc81', [6]), ('Doc85', [0])]
second-> [('Doc81', [33])]
Unique keywords in the document: 18
Magnitude of the document vector: 5.801
Similarity score: 0.228
..........................................................................................
*****************************************************************************************
query: ['company', 'benefit', 'shares']
DID: Doc68
top 5 keywords
benefit-> [('Doc68', [26, 0])]
blender-> [('Doc68', [56, 0])]
company-> [('Doc37', [1]), ('Doc38', [0, 1]), ('Doc40', [1, 2]), ('Doc41', [2]),
('Doc43', [3]), ('Doc45', [4]), ('Doc49', [5, 6]), ('Doc61', [6]), ('Doc65', [7]), ('Doc68',
3
[8, 9, 9, 9]), ('Doc71', [9]), ('Doc76', [10]), ('Doc78', [11]), ('Doc79', [12, 13])]
dean-> [('Doc63', [0]), ('Doc67', [0]), ('Doc68', [1, 2])]
acquisition-> [('Doc35', [21]), ('Doc45', [0]), ('Doc46', [1]), ('Doc47', [2]), ('Doc68',
[3, 4]), ('Doc82', [4]), ('Doc83', [5])]
Unique keywords in the document: 42
Magnitude of the document vector: 10.051
Similarity score: 0.354
..........................................................................................
DID: Doc40
top 5 keywords
computer-> [('Doc39', [0]), ('Doc40', [0, 1]), ('Doc42', [1]), ('Doc43', [2, 3])]
terminal-> [('Doc39', [1]), ('Doc40', [0, 1]), ('Doc42', [1]), ('Doc43', [2, 3])]
company-> [('Doc37', [1]), ('Doc38', [0, 1]), ('Doc40', [1, 2]), ('Doc41', [2]),
('Doc43', [3]), ('Doc45', [4]), ('Doc49', [5, 6]), ('Doc61', [6]), ('Doc65', [7]), ('Doc68',
[8, 9, 9, 9]), ('Doc71', [9]), ('Doc76', [10]), ('Doc78', [11]), ('Doc79', [12, 13])]
share-> [('Doc38', [15, 0]), ('Doc39', [0, 1]), ('Doc40', [1, 2]), ('Doc41', [2]), ('Doc44',
[3]), ('Doc63', [4]), ('Doc65', [5, 6]), ('Doc66', [6]), ('Doc79', [7, 8]), ('Doc80', [8]),
('Doc81', [9, 10, 10]), ('Doc95', [10])]
exercisable-> [('Doc40', [6]), ('Doc41', [0])]
Unique keywords in the document: 28
```

Parthiban Supriya
20567171

```
Magnitude of the document vector: 13.304
Similarity score: 0.123
..........................................................................................
DID: Doc38
top 5 keywords
share-> [('Doc38', [15, 0]), ('Doc39', [0, 1]), ('Doc40', [1, 2]), ('Doc41', [2]), ('Doc44',
[3]), ('Doc63', [4]), ('Doc65', [5, 6]), ('Doc66', [6]), ('Doc79', [7, 8]), ('Doc80', [8]),
('Doc81', [9, 10, 10]), ('Doc95', [10])]
shareholder-> [('Doc38', [17, 0]), ('Doc68', [0])]
april-> [('Doc38', [22, 0]), ('Doc46', [0]), ('Doc71', [1]), ('Doc79', [2]), ('Doc94', [3])]
company-> [('Doc37', [1]), ('Doc38', [0, 1]), ('Doc40', [1, 2]), ('Doc41', [2]),
('Doc43', [3]), ('Doc45', [4]), ('Doc49', [5, 6]), ('Doc61', [6]), ('Doc65', [7]), ('Doc68',
[8, 9, 9, 9]), ('Doc71', [9]), ('Doc76', [10]), ('Doc78', [11]), ('Doc79', [12, 13])]
board-> [('Doc33', [2]), ('Doc38', [0, 1]), ('Doc78', [1])]
Unique keywords in the document: 24
Magnitude of the document vector: 15.187
Similarity score: 0.108
..........................................................................................
******************************************************************************************
query: ['brown', 'forman']
DID: Doc82
top 5 keywords
brown-> [('Doc78', [0, 0]), ('Doc79', [0]), ('Doc80', [1]), ('Doc81', [2]), ('Doc82', [3, 4,
4]), ('Doc83', [4])]
forman-> [('Doc78', [1, 0]), ('Doc79', [0]), ('Doc80', [1]), ('Doc81', [2]),
('Doc82', [3, 4, 4]), ('Doc83', [4])]
corporate-> [('Doc82', [4])]
shearson-> [('Doc82', [24])]
lehman-> [('Doc82', [25])]
4
Unique keywords in the document: 29
Magnitude of the document vector: 10.798
Similarity score: 0.532
..........................................................................................
DID: Doc78
top 5 keywords
brown-> [('Doc78', [0, 0]), ('Doc79', [0]), ('Doc80', [1]), ('Doc81', [2]), ('Doc82', [3, 4,
4]), ('Doc83', [4])]
forman-> [('Doc78', [1, 0]), ('Doc79', [0]), ('Doc80', [1]), ('Doc81', [2]),
('Doc82', [3, 4, 4]), ('Doc83', [4])]
cash-> [('Doc78', [20, 0]), ('Doc79', [0]), ('Doc82', [1])]
dividend-> [('Doc78', [21, 0]), ('Doc79', [0])]
threefortwo-> [('Doc78', [9])]
Unique keywords in the document: 21
Magnitude of the document vector: 16.296
Similarity score: 0.352
..........................................................................................
DID: Doc83
top 5 keywords
formerly-> [('Doc83', [4])]
inc-> [('Doc16', [3]), ('Doc83', [0])]
merger-> [('Doc82', [45]), ('Doc83', [0])]
stearn-> [('Doc75', [12]), ('Doc82', [0]), ('Doc83', [1])]
merrill-> [('Doc24', [15]), ('Doc56', [0]), ('Doc83', [1])]
Unique keywords in the document: 15
Magnitude of the document vector: 17.373
Similarity score: 0.330
```

(d) In the above, we only mentioned about the inverted index. Describe the other data structures you need to maintain to support search and ranking.

Lists – like an array. list can contain any type of objects.

Dictionaries - A dictionary is a sequence of items. Each item is a pair made of a key and a value.

Files, nested lists, nested dictionaries etc.

We use methods like count(), lower(),sorted(), enumerate(),max(), len(),open(),strip(), append() etc.

(e) If the text passages are not updated, how would you design your program to speed up the computation of the similarity values?

If the data set is not updated very often, we can pre-process all the data. But if the data is changing constantly pre-processing can be time consuming and could reduce efficiency.

We can increase the speed of the cosine computation as all document vectors are stored normalized to unit length (or normalisation factors precomputed and stored) i.e. Cosine distance can be quickly calculated as a total series of multiplications. Further, only dimensions in which both elements are non-zero need to be calculated.[4]

I.e.,

$$a \cdot b = \sum_{i \in \mathcal{N}(a) \cap \mathcal{N}(b)} a_i \times b_i$$

where $\mathcal{N}$ returns the indices for the non-zero entries of a vector.

Programming language and version, the OS, and libraries/packages used

Windows 10 64-bit

Anaconda navigator version 5.2.0

Spyder 3.2.8

Python 3- I chose python as the syntax for string operations in Python for coding can be done in fewer steps. It provides large standard libraries for string operations. It is easily readable. and provides an easy usage of the code lines, maintenance can be handled in a great way, and debugging can be done easily too.

Libraries used:

Regex- A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern.

Collections- The Collections module implements high-performance container datatypes (beyond the built-in types list, dict and tuple) and contains many useful data structures that you can use to store information in memory

String- used to perform string operations

Math - The math module gives access to the underlying python library functions

References:

1. https://www.datasciencecentral.com/profiles/blogs/information-retrieval-document-search-using-vector-space-model-in
2. https://docs.python.org/2/tutorial/stdlib.html
3. Stackoverflow..com
4. https://people.eng.unimelb.edu.au/tcohn/comp90042/l4.pdf