INFORMATION RETRIEVAL
ASSIGNMENT

Mrutula suresh
28967828

TOKENIZATION:

The 6 rules are as follows ,

a. Any words hyphenated across a line break must be joined into a single token (with the final token not containing the hyphen).

```
re.findall(r'\b(\w+[-]\n\w+)\b','Hello-\nworld')
['Hello-\nworld']
```

b. Email addresses, web URLs and IP addresses must be preserved as a single token \b(\w+@\w+(?:.com))\b, \b(\d+[.]\d+[.]\d+[.]\d+)\b, \b((?:https:\/\/)?www[.]\w+[.\w]+)\b
Eg msur0003@student.monash.edu, https://gist.github.com, 118.138.100.32

```
In [15]: re.findall(r'\b(\w+@[\w\.]+(?:.com|.edu))\b','msur0003@student.monash.edu')
Out[15]: ['msur0003@student.monash.edu']

re.findall(r'\b(\d+[.]\d+[.]\d+[.]\d+)\b','118.138.100.32')
['118.138.100.32']

In [17]: re.findall(r'\b((?:https:\/\/)?www[.]\w+[.\w]+)\b','www.google.com')
Out[17]: ['www.google.com']
```

c. Text within single quotation marks or inverted commas should be placed in single token
Eg 'hello world' will be captured as a single quote.

```
In [37]: re.findall(r"(?<=[\s,])'(.*?)\.?'(?=[\s,.])","I am writing 'hello world??' bye" )
Out[37]: ['hello world??']
```

d. Two or more words separated by whitespace, all of which begin with a capital letter, must be preserved as a single token

```
|: re.findall(r"(?:[\s,.\"]|^)[A-Z][a-z']+?(?: [A-Z][a-z']+)+(?=[\s\",!.]|$)","Hello World is an Example Prog" )
|: ['Hello World', ' Example Prog']
```

e. Acronym should be preserved as a single token with or without full stop or period [A-Z][.](?:[A-Z][.]?)+
Eg U.S.A is captured

```
In [55]: re.findall(r'[A-Z][.](?:[A-Z][.]?)+',"U.S. is a nation" )
Out[55]: ['U.S.']
```

f. Replacing the following with a space,
   - -- , _,*,=,(,) with space

g. To capture regular tokens,

```
In [57]: re.findall(r'(?:^|[\s,.\'])\"?([\w\-]+)\"?(?=[\s,.!\']|$)'," is a nation" )
Out[57]: ['is', 'a', 'nation']
```

PREPROCESSING

Stopword List : The list provided in this link https://gist.github.com/sebleier/554280 is used as the stop word list for this project.

Length less than 3 letters: After the stop word removal ,words with letters less than 3 are removed and passed to the stemming function

Global word list: Before the words are passed to the stemming block , are stored in a global list for the purpose of spelling correction.

Stemming : For the purpose of stemming , PorterStemmer() from the nltk package

COSINE SIMILARITY:

Indexing : For indexing python dictionary is used

Format : Index = {term :

{Docname: termfrequency

……..,Inversedocumentfrequency : Invdocfreq},……

}

Term Frequency method: It calculates the term frequency for each document

Document Frequency method: This method calculates the document frequency for each term and the inverse document frequency using the formula Log(N/document frequency + 1)

Docvectorize method: This method is used to calculate the TFIDF = TF* IDF for each document . This creates the document vector . This method also calculates and stores the w^2 in the bottom of the cosine similarity formula since its going to be constant for this set of documents.

Queryweight method : This creates the query vectors for the query terms in a dictionary similary to the document vector

Cosinesimilarity method: It calculates the q^2 in the bottom of the cosine similarity formula . If the sum q^2 is 0 then the cosine similarity is returned as 0. Else for each document cosine similarity with the query is found and stored in a dictionary (doc name: cosine similarity value)

Cosine similarity $= \frac{\sum \omega q}{\sqrt{\Sigma \omega^2}\sqrt{\sum q^2}}$ w → document vector q→ query vector

SPELLING CORRECTION:

Isolated spelling correction is performed on the query terms

Edit distance: It has the following algorithm

```
EDITDISTANCE(s1, s2)
 1  int m[i, j] = 0
 2  for i ← 1 to |s1|
 3  do m[i, 0] = i
 4  for j ← 1 to |s2|
 5  do m[0, j] = j
 6  for i ← 1 to |s1|
 7  do for j ← 1 to |s2|
 8       do m[i, j] = min{m[i − 1, j − 1] + if (s1[i] = s2[j]) then 0 else 1fi,
 9                        m[i − 1, j] + 1,
10                        m[i, j − 1] + 1}
11  return m[|s1|, |s2|]
```

Eg: Q: cat and vocabular term : dog

	0	D	1	O	2	G	3
C	1	1		2		3	
A	2	2					
T	3	3					

To change from C to D there is one change happening. From CA to D there is 2 and Cat to D there is 3 . Similarly the first row is filled

	0	D	1	O	2	G	3
C	1	1		2		3	
A	2	2		2		3	
T	3	3		3			

Kepping DO constant to convert to c to DO we have two changes , CA to DO we have two changes and CAT to DO we have 3 changes. Similarly the other rows are filled .

	0	D	1	O	2	G	3
C	1	1		2		3	
A	2	2		2		3	
T	3	3		3		3	

Finally To convert CAT to DOG three changes have to be made . Hence the edit distance for DOG to CAT is 3

k-gram indexes for spelling correction:

When the number of vocabulary terms is large then computing the edit distance between the query and each of the vocabulary term will slow down the system. To limit the number of vocabulary terms considered for edit distance calculation k-gram index is used .

Bi-gram index for a query “bord” Is

bo ,or,rd → three bigrams

Bigrams are calculated for each of the terms in the vocabulary

Eg: aboard → ab,bo,oa,ar,rd

Jaccard coefficient is used to measure the overlap between the two sets.

Jaccard coefficient = $|A \cap B| / |A \cup B|$.

= 2/5 + 3 – 2 = 1/3 = 0.333

A limit of 0.1 is set to select the terms to calculate the edit distance

Spell_check method: Finds the minimum edit distance and appends all the terms with minimum edit distance. If the min edit distance is 0 it means query term is present in the index and no spelling error.

Spelling method:

It displaces the possible options to the user who needs to specify the location of the correct term.

QUERYING:

User types an input. If its q then the program is quit. For each term in the query the spelling is checked using the algorithm. The correct query is then tokenized , pre-processed , query vector generated and then the cosine similarity with the documents is calculated. The top 10 documents in the decreasing order of the cosine similarity is displayed. For each document , the user feedback of whether it is relevant or not is obtained. Based on this the Rocchio algorithm is run which gives modified query vector . This is used to again find the cosine similarity with the documents. The User is given a choice to either quit or see the modified results. He can quit using q or press any other key to view the results. This process is repeated until the user decides to quit using the command q.

```
Enter the query or type q to exit catty
The word wasnt present the vocabulary

['catts', 'catt']
which one did you mean? Choose the position of the word starting from 1
```

User can only choose the position of the correct word . If he enters the word it throws an error

```
which one did you mean? Choose the position of the word starting from 1
catt
User input is string
which one did you mean? Choose the position of the word starting from 1
2
```

ROCCHIO ALGORITHM:

The original query is modified using the following formula

$$\overrightarrow{Q_m} = (a \cdot \overrightarrow{Q_o}) + \left(b \cdot \frac{1}{|D_r|} \cdot \sum_{\overrightarrow{D_j} \in D_r} \overrightarrow{D_j}\right) - \left(c \cdot \frac{1}{|D_{nr}|} \cdot \sum_{\overrightarrow{D_k} \in D_{nr}} \overrightarrow{D_k}\right)$$

$\overrightarrow{Q_o}$ ----> original query vector

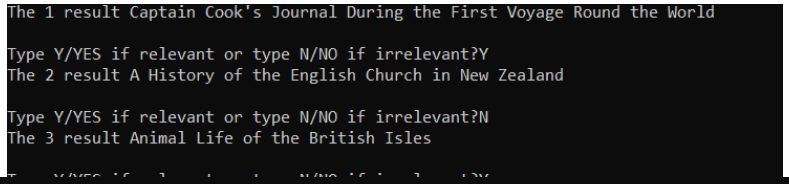
a = 1, b = 0.75, c= 0.25

$\sum_{\overrightarrow{D_j} \in D_r} \overrightarrow{D_j}$ ---> sum of the relevant document vector

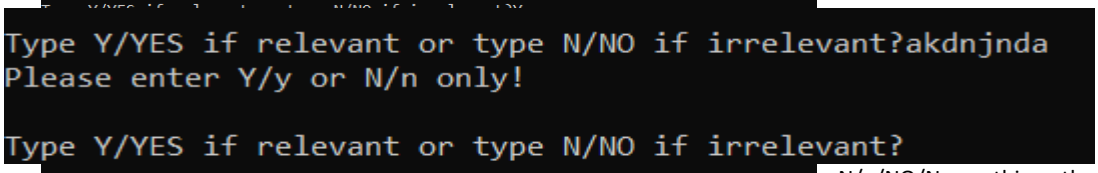
$|D_r|$ ---> sum of relevant documents

$\sum_{\overrightarrow{D_k} \in D_{nr}} \overrightarrow{D_k}$ ---> sum of the non relevant document vector

$\overline{|D_{nr}|}$ ---> sum of non rel doc



User can type only y/Y/Yes/YES or



N/n/NO/No anything other than that will not be accepted

Testing:

Q: what similarity laws must be obeyed when constructing aeroelastic models of heated high speed aircraft

Top 15 Retrieved docs:

51, 746,359,875,56,13,12,486,878,584,879,817,665,253,573,435

Precision@16 = relevant docs/ total retrieved docs = 5/16 = 0.3125

Recall@16 = relevant docs captured by the engine/ total relevant docs = 6/29 = 0.2068