Text Processing University Assignment

Document Retrieval

Introduction:

Wrote a new program in Python for Text Processing & Document Retrieval by implementing Inverted Index for Index the docs and cosine tf_idf algorithm for retrieval. This program uses the given documents.txt, queries.txt and stop_list.txt files.

Implementation:

I have spitted assignment as below sub tasks and implemented the Inverted Index & TF_IDF algorithms.

Document reading

I have utilized the given read_documents.py to read the files. The read_documents.py uses the below Regex to identify the start & end of documents

```
startdoc = re.compile('<document docid\s*=\s*(\d+)\s*>')
enddoc = re.compile('</document\s*>')
```

Tokenization & Preprocessing

Read the doc line by line in a for loop and tokenize or split the words by using default line.split() functionality.

Trim the words and remove the delimiter chars ",.;:!?" by using strip function

```
`word = word.strip(delimiter_chars)
```

Stop list processing

Read the stop list words by using construct the Set in the name stopset. While reading the words, compare the word with stop_set, If it was a not stopset word then used for further processing like keyword. Else ignore those words.

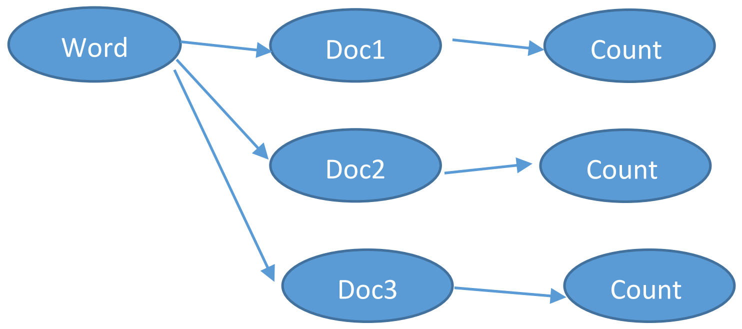
Stemming

I have imported the NLTK module from python to use stemming via SnowballStemmer like below,

```
snowballStemmer=SnowballStemmer("english")
```

Index data Structure

Implemented the inverted index algorithm for indexing. It was similar to Apache Lucene implementation. Please refer below diagram for index structure



Storing and load the Index

Storing the index in below format to database.dat file.

{word1: {doc3: [frequency_word_count_in_doc]}, word2: {doc2: [frequency_word_count_in_doc]}, word3: {doc1: [frequency_word_count_in_doc], doc2: [frequency_word_count_in_doc]}}

Used pickle module to dump the index into database.dat file

```
pickle.dump(newDatabase, fileObject)
```

TF_IDF Matrix building for documents

IDF-Inverse Document Frequency was computed by using below formula

Idf for word = log(N/word_count)

N →Length of docs

TF- Term Frequency. The TF_TDF weighting was computed by

TF_IDF = TF * IDF

So, the word, document and tf_idf values are formed similar to the below table

	Word1	Word2	Word3
Doc1	Tf_idf	Tf_idf	Tf_idf
Doc2	Tf_idf	Tf_idf	Tf_idf
Doc3	Tf_idf	Tf_idf	Tf_idf

TF_IDF Matrix building for queries

The TF_IDF factor for each query word was computed like below

Tf_idf of query word = tf_idf of doc * number of word frequency/ maximum word frequency

Similarity Factor computation and order the results

Compute length of doc & query by using below formula

Length of d1 = sqrt(tf_idf_word1^2+ tf_idf_word2 ^2+ tf_idf_word3^2)
Length of d1 = sqrt(tf_idf_word1^2+ tf_idf_word2 ^2+ tf_idf_word3^2)
Length of q1 = sqrt(tf_idf_word1^2+ tf_idf_word2 ^2+ tf_idf_word3^2)
Then find the similarity values,
cosSim(d1,q1) = (td_ifd_word1_in_d1* td_ifd_word1_in_query1 +…) / (length of d1*lengthq1)
cosSim(d2,q1) = (td_ifd_word1_in_d3* td_ifd_word1_in_query1 +…) / (length of d2*lengthq1)
cosSim(d3,q1) = (td_ifd_word1_in_d3* td_ifd_word1_in_query1 +…) / (length of d3*lengthq1)

Based on the reverse length, the search results are sorted.

Storing the results in file

The results are sorted based on cosSim value in reverse order and stored on Result.txt file. For optimal look and feel in results file, I have stored the first max 10 results per query doc.

```
1  1410
1  2319
1  53
1  3069
1  2371
1  1938
1  2358
1  1642
1  1523
1  69
2  3078
2  2434
2  2863
2  136
```

Command window result & performance timings

```
C:\workspace\Test>Index.py

Document Retrieval Process started...
Program will load the documents, queries and stop_list words from the files
documents.txt, queries.txt and stop_list.txt respectively
Documents Index building: 5.1289999485
Documents TF_IDF processing: 0.704999923706
Query Index building: 0.047000169754
Query TF_IDF processing: 0.0159997940063
Process Query with Documents and Produce results: 0.700999975204
Document Retrieval process completed and the results are stored in "Result.txt"
C:\ workspace\Test>
```