The testing.py script was used for all testing purposes for the results in this file.

1. METADATA AS ZONES AND FIELDS

1.1 ) Description :

-> As described in detail in the README, we used the zone and field data from documents like court name and date posted to tweak the cosine similarity scores for the documents.

-> This metadata fields based ranking can be turned on by setting the variable zones\_metadata\_switch to True in search.py file

-> We had played around with different values of fractional relevance of date and court name for the documents, as well as relevance factors assigned to each court name and based on time gap of the document. By trial and error we had arrived at the best values which are currently put in the search.py file.

1.2 ) Result Statistics :

Q1.TXT

* q1.txt performance with metadata ON =>

Total correct documents to be retrieved = 3

Total correct documents retrieved = 3

Precision = 0.000248015873016

Recall = 1.0

The positions at which the correct documents were retrieved =

600

3792

5419

* q1.txt performance with metadata OFF =>

Total correct documents to be retrieved = 3

Total correct documents retrieved = 3

Precision = 0.000248015873016

Recall = 1.0

The positions at which the correct documents were retrieved =

243

3789

4510

Q2.TXT

* q2.txt performance with metadata ON =>

Total correct documents to be retrieved = 2

Total correct documents retrieved = 2

Precision = 0.000182448458311

Recall = 1.0

The positions at which the correct documents were retrieved =

35

52

* q2.txt performance with metadata OFF =>

Total correct documents to be retrieved = 2

Total correct documents retrieved = 2

Precision = 0.000182448458311

Recall = 1.0

The positions at which the correct documents were retrieved =

1

93

Q3.TXT

* q3.txt performance with metadata ON =>

Total correct documents to be retrieved = 3

Total correct documents retrieved = 3

Precision = 0.000361969111969

Recall = 1.0

The positions at which the correct documents were retrieved =

0

1

3

* q3.txt performance with metadata OFF =>

Total correct documents to be retrieved = 3

Total correct documents retrieved = 3

Precision = 0.000361969111969

Recall = 1.0

The positions at which the correct documents were retrieved =

0

1

3

1.3) Observations:

From the outputs for q1.txt with and and without metadata field based reordering, we observer that while precision and recall remain the same as expected, the positions at which the correct documents are retrieved in our ordering becomes worse when we apply the metadata ordering. The first correct result is obtained at 600th position in the output list when using metadata instead of 243th position when not using metadata.

Similarly in output for q2.txt, the first correct results position slips from 1 to 35 upon using the metadata based reordering. There is no change in position of correct outputs for q3.txt.

1.4) Conclusion

Thus we can conclude that metadata based reordering makes the output worse in terms of pushing the correct document ids further down in the output of the ranked retrieval ordering.

Thus using metadata based reordering harms the results of our system.

1. PSEUDO RELEVANCE FEEDBACK : ROCCHIO FORMULA

2.1) Description :

-> For freetext retrieval, we do a round of pseudo relevance feedback assuming the top 1% of the initially retrieved documents are relevant. We then use Rocchio's formula to generate the expanded query, and then perform another round of cosine similarity computation with this new expanded query to generate the final list of ranked retrieval documents.

-> We had looked for best values to use for alpha, beta and gamma from some research papers and the wikipedia article on rocchio formula and subsequently tested them out. By trial and error, we used the the best values of alpha, beta and gamma that we could find and they are put in freetext\_retrieval.py

2.2) Result Statistics :

Q1.TXT

* q1.txt performance with rocchio ON =>

Total correct documents to be retrieved = 3

Total correct documents retrieved = 3

Precision = 0.000248015873016

Recall = 1.0

The positions at which the correct documents were retrieved =

242

3646

4510

* q1.txt performance with rocchio OFF =>

Total correct documents to be retrieved = 3

Total correct documents retrieved = 3

Precision = 0.000248015873016

Recall = 1.0

The positions at which the correct documents were retrieved =

243

3789

4510

Q2.TXT

* q2.txt performance with rocchio ON =>

Total correct documents to be retrieved = 2

Total correct documents retrieved = 2

Precision = 0.000182448458311

Recall = 1.0

The positions at which the correct documents were retrieved =

1

94

* q2.txt performance with rocchio OFF =>

Total correct documents to be retrieved = 2

Total correct documents retrieved = 2

Precision = 0.000182448458311

Recall = 1.0

The positions at which the correct documents were retrieved =

1

93

Q3.TXT

* q3.txt performance with rocchio ON =>

Total correct documents to be retrieved = 3

Total correct documents retrieved = 3

Precision = 0.000361969111969

Recall = 1.0

The positions at which the correct documents were retrieved =

0

1

3

* q3.txt performance with rocchio OFF =>

Total correct documents to be retrieved = 3

Total correct documents retrieved = 3

Precision = 0.000361969111969

Recall = 1.0

The positions at which the correct documents were retrieved =

0

1

3

2.3) Observations:

From the outputs for q1.txt, we can observe that precision and recall remain the same whether we use the rochio based pseudo relevance feedback or not. We can also observe that while positions for the first two correctly retrieved document in the output is marginally improved from 243rd to 242nd position and from 3789th to 3646th position respectively, and the position of third correctly retrieved document remained the same at 4510th position.

Similarly, in q2.txt the precision and recall remains same in either case but the position of the second correctly retrieved document drops slightly from 93rd to 94th position.

In q3.txt, the positions of the correctly retrieved documents remains the same whether we use the rochio based pseudo relevance feedback or not.

2.4) Conclusion

Thus we can conclude that rocchio base pseudo relevance feedback leads to marginal improvements in some cases while marginal decline in other cases, in the output of the ranked retrieval based ordering of documents. Thus using rocchio base pseudo relevance feedback does not lead to significant improvement as a query expansion technique.

1. SYNONYMS : WORDNET-QUERY EXPANSIONS

3.1 ) Description :

-> As described in detail in the README, if the query contains a term that does not exist in our dictionary, we use WordNet to search for a single synonym that exist in our dictionary to replace the original term.

-> This synonym handling can be turned on by setting the variable synonyms\_switch to True in synonyms.py file

3.2 ) Result Statistics :

Query used -> cantillate

* performance with synonyms ON =>

Total documents retrieved = 5

* performance with synonyms OFF =>

Total documents retrieved = 0

3.3) Observations:

Cantillate is a word which is not there in the dataset.csv corpus provided to us. Thus it isn’t present in the dictionary either. From the outputs for the query with and and without synonym handling, we observe that the word 'cantillate' returns nothing when the switch is OFF, but returns 5 documents when we handle synonyms. When we turn the switch for synonyms ON, the word cantillate is substituted with the word ‘chant’ by the WordNet which is indeed present in the dictionary and thus we are able to fetch the 5 results corresponding to the word chant, which is a synonym for the word cantillate.

3.4) Conclusion

We can conclude that handling synonyms can help retrieving documents for queries where the terms aren’t present in the corpus at all, by instead substituting them with synonyms of the term which are present in our dictionary and thus preventing empty outputs. Thus using synonyms is beneficial to the results of our system.

1. SYNONYMS : THESAURUS

4.1) Description :

-> As described in detail in the README, if we want to expand the query, we can choose to expand it by a list of very similar terms in the corpus itself. This is an alternative to the get\_synonyms method described above, which can be used to exploit the similarity between any two terms in the corpus.

-> This thesaurus handling can be turned off by setting thesaurus\_switch to be False in the file, freetext\_retrieval.py

4.2) Result statistics:

Q1.TXT

* q1.txt performance with thesaurus\_switch True =>

Total correct documents to be retrieved = 3

Total correct documents retrieved = 3

Precision = 0.000179662234998

Recall = 1.0

The positions at which the correct documents were retrieved =

1076

3608

10679

* q1.txt performance with thesaurus\_switch False =>

Total correct documents to be retrieved = 3

Total correct documents retrieved = 3

Precision = 0.000248015873016

Recall = 1.0

The positions at which the correct documents were retrieved =

243

3789

4510

Q2.TXT

* q2.txt performance with thesaurus\_switch True =>

Total correct documents to be retrieved = 2

Total correct documents retrieved = 2

Precision = 0.000117924528302

Recall = 1.0

The positions at which the correct documents were retrieved =

27

102

* q2.txt performance with thesaurus\_switch False =>

Total correct documents to be retrieved = 2

Total correct documents retrieved = 2

Precision = 0.000182448458311

Recall = 1.0

The positions at which the correct documents were retrieved =

1

93

Q3.TXT

* q3.txt performance with thesaurus\_switch True =>

Total correct documents to be retrieved = 3

Total correct documents retrieved = 3

Precision = 0.000194666147557

Recall = 1.0

The positions at which the correct documents were retrieved =

0

1

3

* q3.txt performance with thesaurus\_switch False =>

Total correct documents to be retrieved = 3

Total correct documents retrieved = 3

Precision = 0.000361969111969

Recall = 1.0

The positions at which the correct documents were retrieved =

0

1

3

1.3) Observations:

From the outputs for q1.txt with and and without thesaurus based expansion, we observer that while recall remains the same as expected, the precision decreases and the positions at which the correct documents are retrieved in our ordering becomes worse when we apply the thesaurus based expansion. The first correct result is obtained at 1076th position in the output list when using thesaurus instead of 243th position when not using thesaurus.

Similarly in output for q2.txt, the first correct results position slips from 1 to 27 upon using the thesaurus based expansion, while for one of the relevant documents, the retrieved position improves. There is no change in position of correct outputs for q3.txt, because it is based on pure positional indexing. However, the precision decreases for each of the cases, though recall remains the same and the ordering of the relevant documents also falls compared to original

1.4) Conclusion

Thus we can conclude that thesaurus based expansion makes the output worse in terms of pushing the correct document IDs further down in the output of the ranked retrieval ordering.

Thus using thesaurus based expansion harms the overall F-score of our ranked-retrieval system.