Username: acc15jc

Document retrieval report

In each term weighting method (binary, tf and tfidf), documents are initially retrieved by matching the terms in precompiled index file with the keywords in the query. Then the scores are computed to rank those documents using the following methods.

A dictionary called *doc_scores* is used to store documents and their scores.

Binary: Each document with term presented is put into the *doc_scores* with score of **-1** initially. Then if document has another term in the query, its score will increment **-1**, which means it is more relevant to the query. Here the values are **negative** such that the sorting at the end will not need to be reversed (as it arranges from small value to large value).

Vector space model is used in Tf and Tf.idf method.

Tf (Term frequency): Firstly, the vector of query is created by simply mapping its term frequency of every keyword into a list. Then for every different document, it is put in the *doc_scores* with a list of 0 with same size as the query list initially. Next, the document's term frequency is put into the list at keyword-corresponding place by enumerating the query. The values in *doc_scores* are temporally lists representing the vector of document, at the end those lists will be replaced with the scores as the cosine distances between query vector and document vectors. The scipy.spatial library is used here to compute the distance easily. Notice that the values are cosine **distances** (1 - cosine value), it is proportional to the angle between document vector and query vector. And smaller angles are wanted from the documents, such that the sorting at the end will not need to be reversed.

Tf.idf (inverse document frequency): At the beginning, the number of documents is obtained by putting all documents into a set, (same documents will only appear once) then get the length of this set. Next, the documents vectors and the query vector are constructed, same as in Tf method, but difference is that tf.idf values are computed (math.log(num_of_docs/document_frequency, 10)) and assigned into either query or documents vectors. Finally, the cosine similarity distance is calculated, same as in Tf method.

At last, the *doc_scores* dictionary is sorted into a list of documents according to the score and the ranked documents are returned to ir_engine.

Below are the results of each ranking method:

Table 1: Results of each ranking methods applying in files with different retrieving methods.

Binary	No stop, no stem	with stop, no stem	No stop, with stem	With stop, with stem
retrieved	640	640	640	640
relevant	796	796	796	796
Rel_retrieved	67	85	74	95
precision	0.10	0.13	0.12	0.15
Recall	0.08	0.11	0.09	0.12
F-measure	0.09	0.12	0.10	0.13
----------	----------	----------	----------	----------
Tf	No stop, no stem	with stop, no stem	No stop, with stem	With stop, with stem
retrieved	640	640	640	640
relevant	796	796	796	796

Username: acc15jc

Rel_retrieved	60	88	74	103
precision	0.09	0.14	0.12	0.16
Recall	0.08	0.11	0.09	0.13
F-measure	0.08	0.12	0.10	0.14
----------	----------	----------	----------	----------
Tf.idf	No stop, no stem	with stop, no stem	No stop, with stem	With stop, with stem
retrieved	640	640	640	640
relevant	796	796	796	796
Rel_retrieved	120	121	128	121
precision	0.19	0.19	0.20	0.19
Recall	0.15	0.15	0.16	0.15
F-measure	0.17	0.17	0.18	0.17

As result, vertically comparing, **tf.idf** is better than **tf** which is better than **binary** method. Horizontally viewing, in binary method and tf method, the order of documents retrieving method from best to worst is **with stop list and steming**, **only with stop list**, **only with steming** and **without any**. However, in the tf.idf method, scores are very similar among retrieving methods, **only with steming** is slightly better than the other 3 methods. This could be explained as the idf (inverse document frequency) has already taken care of the common but not useful terms, however the stop list may consider those terms has the same level of 'unusefulness' which could cause some useful information lost.

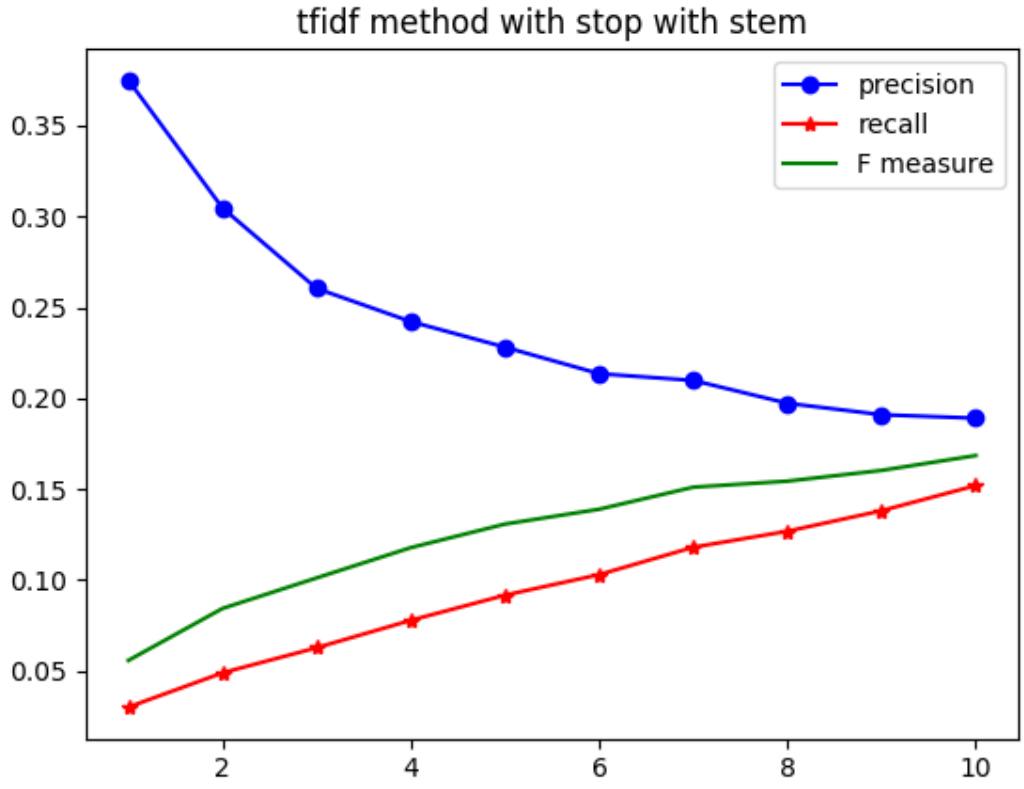

Figure 1: number of documents considered against the score using tf.idf method in stop and stem file.

This figure tells us that precision can be increased by retrieving less ranked documents (38% as highest), but it will lower the recall score and thus the F measure score.