IR ASSIGNMENT-2

GROUP- 19

SHUBHAM PREMPRAKASH SHUKLA	2017A2PS0821P
SUBHAM KUMAR DASH	2017A7PS0004P
DEEPAK CHAHAR	2017A7PS0147P

ALGORITHM FOLLOWED:

1) First, we parsed the entire corpus and made a dictionary of dictionaries, as shown below:

{ (doc1_id : {(term1: frequency), (term2: frequency)…(last_term_in_doc1)}), (doc2_id : {(term1: frequency), (term2: frequency)…, (last_term_in_doc2: frequency)}) }

Another dictionary is made as follows:

{(term1:[doc_id's which contain this term]), (term2: [doc_id's which contain this term])…}. This dictionary is made for all the terms in the corpus.

2) Then, the input is taken from the user, it is tokenized. The tf score is found for the query terms. For calculating the idf score in query, the dictionary created above is used. Suppose for a term t in query, the no. of docs which contain that terms is just the length of the list, which is the value in the (key, value) pair, in the the 2nd dicitionary.

   Then, the query terms are cosine normalized.

3) The tf score of all the terms in a particular doc if found out from the dictionary which was created first. Then, these scores are cosine normalized.
4) In theory, all the docs and the query terms are projected as unit vectors in N-dimensional space (where N is the length of the corpus). Then, the docs which have least angle(maximum cosine value), are retrieved in descending order of scores. This is found by taking a product of the cosine normalized scores of the corresponding terms in the query and the docs.

Assumptions:

1) It is a Bag of words model. No positional importance is given.
2) It is assumed that all the terms can occur independent of each other in the docs.
3) **We assume that the relevance of query is directly proportional to the query-document similarity.**

Limitation:

1) Assumes that all the terms are totally independent of each other . Like collocations.
2) The documents which have the same information but different vocabulary are considered very different from one another. But, that should not have been the case.
3) The individual query terms must exactly match the document terms. Hence, results in many false positives.
4) Positional importance of words in the docs are completely lost.

Query : 1 Novel corona virus	Relevant docs (top 10)	Score	Is the document relevant or not?
	5552222	0.082611	No
	5550931	0.076884	No
	5566343	0.060676	No
	5566192	0.059817	No
	5551093	0.058636	No
	5566334	0.055225	No
	5575139	0.052792	No
	5566179	0.051523	No
	5548530	0.051289	No
	5574100	0.050788	No

Query : 2 Best friends	Relevant docs (top 10)	Score	Is the document relevant or not?
	5567414	0.115305	Yes
	5555746	0.112345	Yes
	5566106	0.109087	Yes
	5574930	0.108676	Yes
	5568655	0.098823	Yes
	5560016	0.097973	Yes
	5561166	0.097846	Yes
	5549900	0.093306	Yes
	5552590	0.090407	Yes
	5563647	0.062492	Yes

Query : 3 Slovenian under-16 and under-19 national team	Relevant docs (top 10)	Score	Is the document relevant or not?
	5571205	0.150545	Yes
	5571147	0.114121	Yes
	5573079	0.075982	Yes
	5571117	0.074787	Yes
	5572655	0.068413	No
	5555678	0.065166	No
	5555607	0.062749	No
	5570182	0.061392	No
	5570616	0.005739	No
	5554162	0.052463	No

This is because there were only 4 docs (which came on top 4 in our result), which contained anything about the Slovenian national team. Then, the model is just picking on the individual terms.

Query : 4 Trans-genders of the united states of America	Relevant docs (top 10)	Score	Is the document relevant or not?
	5570760	0.289781	No
	5557658	0.288396	No
	5563293	0.288396	No
	5574155	0.288396	No
	5548514	0.26796	No
	5548714	0.26796	No
	5557646	0.26796	No
	5549667	0.251340	No
	5574455	0.199176	No
	5569406	0.186885	No

The query does not have any related docs in the database. The model is just returning docs on the basis of the individual terms , “united”, “states”, “America”. That’s why most scores are almost the same.

Query :5 Director of prison reform trust	Relevant docs (top 10)	Score	Is the document relevant or not?
	5549019	0.458747	Yes
	5548734	0.224864	Yes
	5574092	0.147367	No
	5572608	0.142115	No
	5563084	0.086498	No
	5561633	0.085170	No
	5551832	0.074835	No
	5565110	0.069648	No
	5565925	0.066522	No
	5572158	0.064199	No

The first doc contained the exact statement. Hence, such a marginal difference in the scores.

There were only 2 docs on prison reform trust, both of which are retrieved on the top. Rest, the model is just picking up on the individual terms.

Query : 6 Enchantment February 6, 1937	Relevant docs (top 10)	Score	Is the document relevant or not?
	5552500	0.140033	Yes
	5556013	0.087862	Yes
	5553627	0.075990	yes
	5562506	0.072137	Yes
	5556381	0.058366	Yes
	5572852	0.055049	Yes
	5553360	0.054943	Yes
	5573417	0.054698	Yes
	5549115	0.052930	Yes
	5550187	0.051484	Yes

Here all the values of doc_id came by giving weightage to February. Very less weightage to numbers because they are more common. This can be improved using bi-word indexes.

The model is retrieving anything which happened in the month of February in the database.

Query : 7 To be or not to be	Relevant docs (top 10)	Score	Is the document relevant or not?
	5572912	0.333001	No
	5557083	0.314039	NO
	5563605	0.304665	No
	5556723	0.295397	No
	5551771	0.265787	No
	5568606	0.260095	No
	5555774	0.255443	No
	5562507	0.254487	No
	5553242	0.248362	No
	5564044	0.248165	No

This is a very absurd query. The scores are almost the same because they are totally based on individual terms.

Query : 8 Most valuable football team in china	Relevant docs (top 10)	Score	Is the document relevant or not?
	5574605	0.151589	Yes
	5572764	0.146308	Yes
	5565270	0.145923	Yes
	5565187	0.140692	Yes
	5557389	0.119918	Yes
	5571147	0.115057	Yes
	5573254	0.112992	Yes
	5551103	0.111429	Yes
	5558012	0.111429	Yes
	5559701	0.110017	Yes

The doc which comes on the 5th in this list is the most relevant as it contains the exact query statement. Yet, other documents have higher weightage because they have very high term frequency of the term “football”.

This can be improved by using bi-word indexes.

Query :9 European politics vs Belgian politics	Relevant docs (top 10)	Score	Is the document relevant or not?
	5556863	0.129542	Yes
	5557095	0.115906	Yes
	5557589	0.114422	Yes
	5571555	0.111801	Yes
	5569861	0.096679	Yes
	5564650	0.089098	Yes
	5552962	0.081436	Yes
	5562651	0.076893	Yes
	5571658	0.076447	Yes
	5552801	0.072617	Yes

Very generic query

Query : 10 Show me erotic Japanese paintings	Relevant docs (top 10)	Score	Is the document relevant or not?
	5563249	0.179290	No
	5564271	0.121538	No
	5570721	0.107878	No
	5557263	0.097793	Yes
	5569974	0.083026	No
	5563297	0.079946	No
	5562735	0.073798	No
	5572994	0.073280	No
	5554643	0.067707	No
	5570665	0.064994	No

Most searched query on the internet!!!

The 4th document is the most relevant document, but the top 3 documents have a higher term frequency of the word “Japanese”, which is a rare term.

Creating bi-word indexes gave the most relevant documents on the top(we’ve checked it!).

This type of information is present nowhere in the corpus. Hence, the documents returned are based on individual terms only.

Improvement: 1

1. The previous implementation does not involve lemmatization. Suppose, in the query, we want to find some term which is in the base form, but in some document which is relevant(contextually), the same query term is present in some non-lemmatized form, then the model would consider those two terms to be different. But, it should consider them to be the same.

2. Lemmatize the terms of both the documents(and storing the lemmatized form in the inverted index construction)and the query terms.

3. Now both the query term and the document terms which mean the same, will be in the same root form.

4. Suppose the user wants to find the exact query statement. Then, lemmatizing the query may lead to some loss of information. E.g. The user wants to specifically find the query, “My Best friend”. But, this query gets lemmatized to “My good friend”. So, there will be noise in the information retrieved by the system.

5. **Query 1: Good friend/Best friend**

Earlier(when using only uni-words) when we searched for the query ”Best friend”, it gave the result for only “Best friend”. But, after lemmatization, the same query will yield result which match both “good friend” as well as “best friend”. Thus, it increases the context

Query 2: Care of a mother/ Caring mother

Both the queries have the same context, and so the retrieved docs must also be the same. This is done through lemmatization

Query 3: Ram is swimming/ Ram swim’s

Both swim and swimming have the base form as swim. So, both the queries will return the same docs, which is correct because they have the similar context.

Improvement: 2

1. Basically, the original model is ‘A BAG OF WORDS’ model. It gives absolutely no concern to positional importance. But, some collocations which appear in the query, should be searched in pairs only, in the documents also.

2. We are making Bi-word dictionaries, as well as extended Bi-word Dictionaires, along with using the previously made dictionary.

3.Inverted index are basically made using the uni-word and the bi-word. Then, the scorings are calculated first using the bi-words. If, the user defined no. of documents (K) are retrieved using the bi-word index, then only those are displayed as relevant docs. If not sufficient no. of documents are retrieved using the byword indexes, then the remaining docs are retrieved using the uni-word indexes, because byword indexes give importance to positions. Hence, they are given priority.

4.

5. **Query: February 6, 1937.**

Earlier, all the docs retrieved contained February and none of them contained 6 and 1937. This was because, 6 and 1937 are relatively frequent in the docs but February Is not. Hence more importance to February. Even the document containing this exact document came at 4th place, because some other document had much higher term frequency of February. But in the second model, the system (February, 6), (6, 1937). Using this the document which actually contained this doc came at the top

Query: **Show me Erotic Japanese paintings.**

There was a document which directly contained "Erotic Japanese paintings". But, it Never made it to the top K in the earlier model (because there was another model which had high term frequency of Japanese). But, in the second model, This Document came at the top.

Query: **New Liberty Square**

It's a name of a place. In the previous model, all the terms were search Independently. So, the most relevant document(manually found), came at the 2nd position. But in the Byword model, these terms were found out together, hence the most efficient model came at the 1st position.