CS6370 - NLP - Project Report

Adepu Vaishnavi, Rishabh Singh Gaharwar, and Rahul Bhateja

Indian Institute of Technology, Madras

Abstract. Information Retrieval (IR) is a fundamental task with heavy applications of natural language processing (NLP), aiming to efficiently retrieve relevant information from large datasets. Here, we implement and compare various techniques used in IR systems. We employ Vector Space Model(VSM), Latent Semantic Analysis(LSA) , nGram IR and variants of Latent Semantic Analysis - Normalized LSA and Bigram LSA hybrid.

Keywords: Information Retrieval, Natural Language Processing, Vector Space Model, Latent Semantic Analysis, nGram, Query Expansion(QE)

1 Introduction

Information Retrieval (IR) is the task of identifying and retrieving information resources that are relevant to an information need. It is a vital component of various applications, including web search engines, digital libraries, and recommendation systems. In this project, we have focused on retrieval of relevant documents for a given query. We have used the Cranfield dataset as the collection of documents. Cranfield dataset is one of the most widely used corpus for testing IR systems and is considered a standard benchmark in the field of information retrieval. It consists of a set of 1400 *abstracts* from documents related to aeronautics and a set of 225 *queries*, also related to aeronautics. Each document-query pair has a *relevance score* obtained from human knowledge.

We have implemented multiple commonly used techniques, including a Vector Space Model, Latent Semantic Analysis (LSA), Normalized LSA, nGram IR. We perform hypothesis testing and draw conclusions from the results we have obtained on evaluating our implementation. We perform hypothesis testing to compare different techniques under different evaluation measures in IR under different assumptions.

2 Definitions

2.1 Vector Space Model

Vector Space Model is a mathematical framework used in IR and NLP to represent documents and queries as vector in a high-dimensional space. Documents and queries are typically represented as term frequency(tf) * inverse document frequency(idf) vectors, where each dimension corresponds to a unique term in corpus.

2.2 nGram IR

nGram refers to a contiguous sequence of n words from a given text. In context of IR, nGram IR models documents and queries by decomposing all sentences into nGrams. nGrams are helpful in capturing word order in text, which is a limitation of vector-space representation of text.

2.3 Latent Semantic Analysis

LSA is a technique used in NLP and IR to analyze the relationships between a set of documents and the terms they contain by producing set of underlying concepts related to documents and terms. LSA aims to uncover the latent (hidden) structure in a collection of texts. We intend to handle synonym and to an extent polysemy using LSA.

2.4 Query Expansion

In ***Query Expansion*** we intend to expand the query by appending additional terms that are synonymous to the original terms present in the query. We use Wordnet synsets to do so.
To give an example - Query 'What is a car?' becomes 'What is a car automobile?' and so on.

2.5 Hypothesis Testing

Hypothesis testing plays an important role in differentiating truly better techniques of IR from the rest. Just looking at the results obtained is not sufficient, as some techniques can by chance perform better given the set of topics, evaluation metric etc. In this project, we have considered hypothesis testing with our domain as Cranfield Dataset and evluation metric as nDCG. Assumptions vary between the models being compared. We use Student's t-test for hypothesis testing. Our *null hypothesis* is that there is no difference between the models being compared. This null hypothesis needs a two-tailed test and we fix $\alpha = 0.05$. If the p_value obtained is smaller than α, we reject our null hypothesis. If not, then our null hypothesis can not be rejected.

3 PreProcessing

3.1 Sentence Segmentation

We divide given document or query into sentences i.e, representing them as lists of sentences. This is done using the 'Punkt Sentence Tokenizer' from nltk library.

3.2 Tokenization

We divide each sentence into smaller meaningful units called tokens for further preprocessing. This is done using 'Penn Treebank Tokenizer' from NLTK library.

3.3 Inflection Reduction

We can either use Stemming or Lemmatization for this, we have decided to go with lemmatization as it finds meaningful word/ representation. As computation power is not an issue since we are not dealing with a huge corpus, using lemmatization is appropriate considering the drawbacks of stemming. This is done using the Lemmatizer present in the NLTK library.

3.4 Stop Words Removal

We have have used NLTK's collection of stopwords to filter out any stopwords from the preprocessed corpus and queries.

3.5 Spell Check

Additionally we have implemented spell check into the preprocessing function, using the `autocorrect` library. The reason for choosing `autocorrect` was the better performance received on custome queries compared to `edit_distance` and `jaccard_index` on bigrams. To give an example: Given an input query *a smal weng plane* which a mistyped query with the intended query being *a small wing plane*, the following corrections were obtained from the techniques used:

- `edit_distance` - *a sal wang plane*
- `jaccard_index` - *a small wen plane*
- `autocorrect` - *a small went plane*

4 Evaluation Measures

We have explained evaluation measures in depth in the NLP_Short file. Kindly refer to that to learn of the evaluation measures used in this project.

5 Vector Space Model

Below five levels of preprocessing is done for all IR methods mentioned -

- Sentence Segmentation
- Tokenization
- Lemmatization
- Stop word Removal
- Spell Check

Implementation details for VSM are clearly stated in the NLP-Short file along with code snippets. We skip explaining the procedure again.

5.1 Time Taken

Time taken for VSM IR to run is 28.993536949157715 seconds

5.2 Results

k	Precision	Recall	$F_1 - score$	$F_{0.5} - score$	MAP	nDCG
1	0.635	0.108	0.178	0.300	0.635	0.511
2	0.544	0.180	0.254	0.356	0.684	0.474
3	0.477	0.225	0.284	0.360	0.691	0.450
4	0.421	0.257	0.296	0.348	0.690	0.435
5	0.379	0.285	0.300	0.333	0.681	0.428
6	0.353	0.314	0.307	0.325	0.671	0.430
7	0.330	0.339	0.309	0.315	0.664	0.434
8	0.315	0.364	0.312	0.308	0.650	0.442
9	0.296	0.380	0.308	0.296	0.646	0.445
10	0.277	0.393	0.301	0.281	0.634	0.448

Table 1. Performance metrics for different values of k(VSM)

The plot we obtained is as below (using $F_{0.5}score$)-

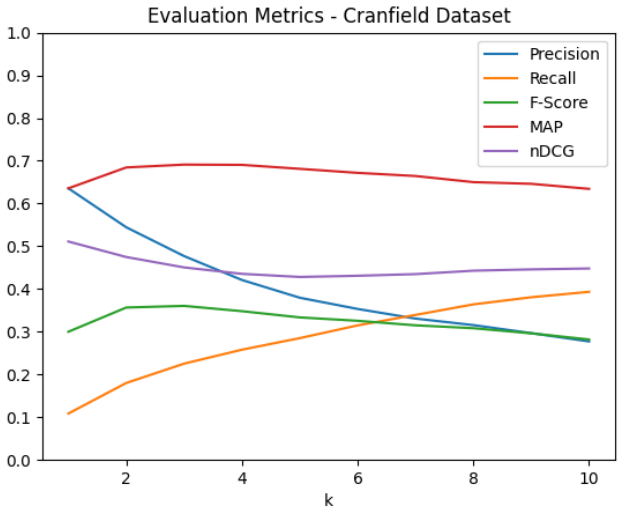

Fig. 1. Performance metrics plot for different values of k (VSM)

5.3 Limitations

- **Bag of words Assumption** - VSM treats each document as a bag of words, disregarding the order of words and their relationships within the document. Also, the bag of words model is built upon the orthogonality assumption, which assumes independence between each pair of words, which is not a valid assumption.
- **Synonym** - In VSM, synonyms pose a significant drawback because they are treated as separate terms, leading to potential issues in capturing the semantic similarity between words.Hence, each unique term in the document collection forms a dimension in the vector space, two similar but distinct words will end up having orthogonal components.
- **Polysemy** - In VSM, polysemous words are represented as a single term despite having different meanings. This can generate false results.
- **Dimensionality** - Increase in dimensions leads to sparsity in data, and since measuring cosine similarity is at the heart of VSM due to high dimensionality similarity measure won't be accurate. In VSM, the number of dimensions is the total number of unique tokens present which is huge.

6 VSM + Query Expansion

In order to combat the limitation of synonym highlighted above we use Query Expansion, where in for a query we append additional terms that are synonymous to the original terms present in the query.

6.1 Query Expansion in Detail

For each term in query we retrieve its synsets from wordnet and append the name of the first synset to the query. An example of this is mentioned in the definitions. We've have additionally noticed that there are words like 'high-speed' in queries and for term like those wordnet will return no synsets hence the word is broken up into two parts that are 'high' and 'speed' and the name of the first synset for both the words are appended.

6.2 Time Taken

Time taken by VSM + QE IR is 53.5964572429657 seconds

6.3 Results

k	Precision	Recall	$F_{0.5}$-score	MAP	nDCG
1	0.627	0.108	0.297	0.627	0.504
2	0.536	0.177	0.351	0.684	0.470
3	0.470	0.226	0.358	0.688	0.448
4	0.418	0.266	0.350	0.686	0.436
5	0.383	0.295	0.339	0.680	0.431
6	0.348	0.313	0.320	0.672	0.430
7	0.320	0.331	0.304	0.664	0.430
8	0.304	0.355	0.297	0.652	0.435
9	0.287	0.374	0.287	0.639	0.440
10	0.270	0.388	0.275	0.627	0.442

Table 2. Performance metrics for different values of k(VSM+QE)

The plot obtained is as follows-

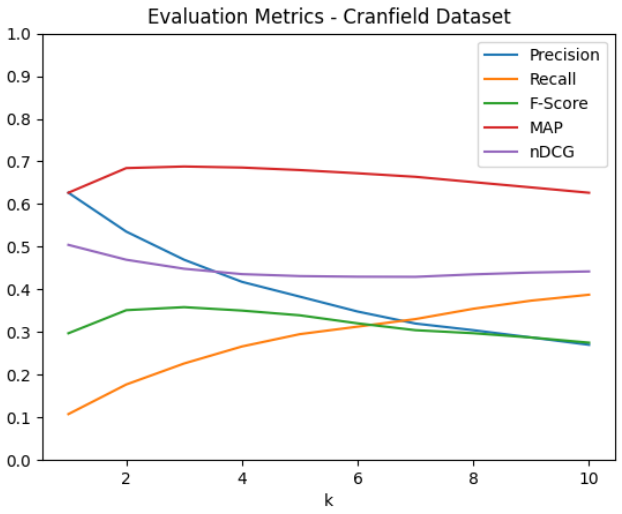

Fig. 2. Performance metrics plot for different values of k (VSM+QE)

6.4 Hypothesis Testing for VSM + QE vs VSM

We check whether for this dataset VSM + QE does perform any different than VSM alone using hypothesis testing.

null hypothesis: - VSM with Query Expansion is not significantly better than VSM with respect to evaluation measure nDCG in Information Retrieval on Cranfield Dataset under Bag of Words assumption

On using the nDCG values obtained for VSM and VSM with Query Expansion, using two tailed t-test, we get $p\_value =$ `0.0201`. Since p_value is less than α, null hypothesis is rejected.

Therefore, we can say that *VSM performs better than VSM with Query Expansion with respect to nDCG in Information Retrieval on Cranfield dataset under bag of words assumption.*

Reasons as to why VSM+QE is performing worse than VSM:

- It is probable that synonyms of a word occur in the corpus, but not within individual documents.
- This means that the dimensionality of query is being increased, which in turn increases the magnitude of the query.
- However, the dot product of the augmented query with originally relevant document remains the same, as it is unlikely for synonyms of a word to appear in the same document.
- Since magnitude of the query increases, similarity score decreases, leading to worse results.
- It is also possible that synonyms of a word are present in documents that are not relevant to the original query. The query augmentation causes the IR system to retrieve those documents as well, reducing performance.
- Another possibility is that a synonym of a certain word is polysemous. This polysemous synonym might be present in multiple documents with different context. This would also decrease the performance of the IR system.

6.5 Limitations

- The bag of words assumption still holds in this case.
- High Dimensioanlity persists as we've not made any efforts in reducing the dimensions
- Polysemy is still a huge drawback of VSM + QE.
- The problem of synonymy has not been effectively addressed in the system, which may lead to low recall.

7 nGram IR

To combat the bag of words assumption we propose a nGram approach, earlier our dimensions were represented by a single token but in this each dimension corresponds to a nGram. Varying this nGram's length i.e, n would also help us deal with dimensionality.
Since n can be any value between 1 and length of the shortest document or query. Choosing a high n is not advisable as it is very rare for two documents to

have the same exact sequence of term, hence choosing a appropriate n is crucial. We take two popularly taken values for n that are $2, 3$ and comment on their relative performance and the best among the two with VSM IR's performance.

7.1 Bi-gram IR

Time Taken Time Taken for bi-gram IR to run is 531.015287399292 seconds.

Results -

k	Precision	Recall	$F_{0.5}$-score	MAP	nDCG
1	0.547	0.092	0.254	0.547	0.451
2	0.458	0.148	0.298	0.607	0.411
3	0.387	0.186	0.296	0.613	0.384
4	0.347	0.218	0.290	0.614	0.375
5	0.310	0.242	0.276	0.603	0.369
6	0.285	0.261	0.265	0.589	0.369
7	0.261	0.279	0.252	0.581	0.367
8	0.238	0.289	0.235	0.577	0.366
9	0.217	0.295	0.219	0.574	0.364
10	0.208	0.312	0.214	0.566	0.370

Table 3. Performance metrics for different values of k(Bi-gram IR)

Plot obtained is -

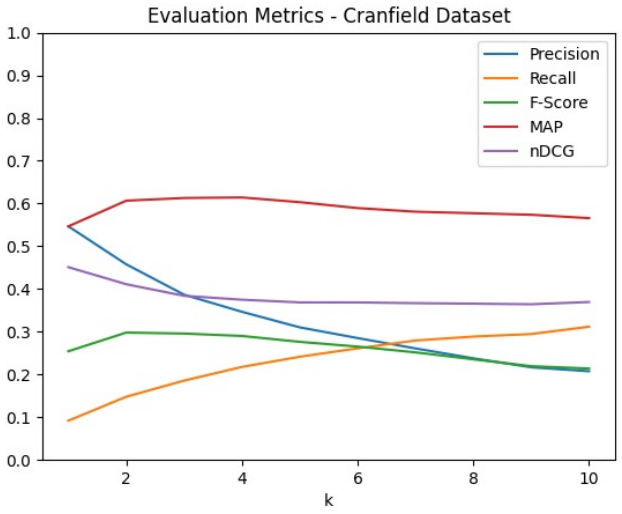

Fig. 3. Performance metrics plot for different values of k (Bi-gram IR)

7.2 Tri-gram IR

Time Taken Time taken for Tri-gram IR to run is 697.7115108966827 seconds.
Results -

k	Precision	Recall	$F_{0.5}$-score	MAP	nDCG
1	0.382	0.063	0.177	0.382	0.317
2	0.300	0.094	0.193	0.424	0.279
3	0.243	0.112	0.183	0.426	0.254
4	0.198	0.120	0.164	0.422	0.237
5	0.171	0.129	0.151	0.418	0.227
6	0.150	0.135	0.139	0.417	0.221
7	0.133	0.138	0.127	0.418	0.217
8	0.120	0.142	0.118	0.416	0.215
9	0.110	0.145	0.111	0.415	0.213
10	0.100	0.147	0.103	0.413	0.213

Table 4. Performance metrics for different values of k(Tri-gram IR)

Plot is as follows-

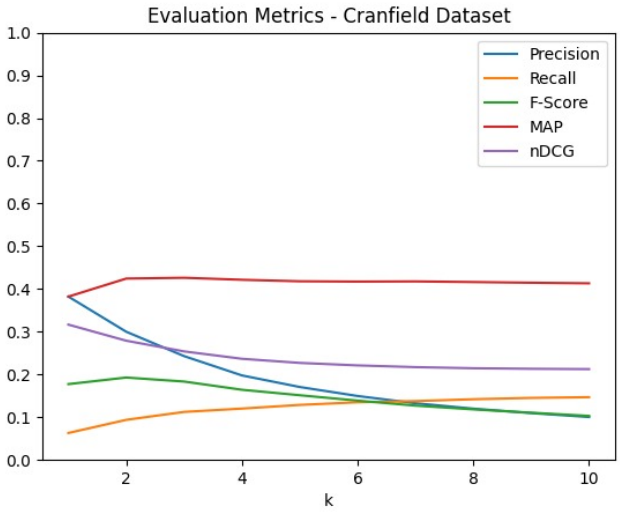

Fig. 4. Performance metrics plot for different values of k (Tri-gram IR)

7.3 Hypothesis Testing for Bi-gram vs Tri-gram IR

We check whether for this dataset, Bi-Gram IR performs any better than Tri-Gram IR using hypothesis testing.

***null hypothesis*:** - Bi-Gram IR is not significantly better than Tri-Gram IR with respect to evaluation measure nDCG in Information Retrieval on Cranfield dataset under Sequential Dependence Assumption.

On using the nDCG values obtained for Bi-Gram IR and Tri-Gram IR using two-tailed t-test, we get $p\_value = 3.54 \times 10^{-12}$. Since p_value is less than α, null hypothesis is rejected.

Therefore, we can say that *Bi-Gram IR performs better than Tri-Gram IR with respect to nDCG in Information Retrieval on Cranfield dataset under Sequential Dependence Assumption*

nGram IR Systems assume that order of words matters for understanding the context and meaning of the text.This is the Sequential Dependence Assumption.

Reasons as to why bi-gram IR performs better than tri-gram IR

- Tri-grams are more sparse than bi-grams. The probability of three words occurring in the same order is lesser than the probability of two words occurring together. This leads to poorer results.

7.4 Hypothesis Testing for nGram IR vs VSM

Since bi-gram performs better than tri-gram we perform hypothesis testing for bi-gram IR vs VSM.

***null hypothesis*:-** Bi-Gram IR is not significantly better than VSM with respect to evaluation measure nDCG in Information Retrieval on Cranfield dataset under Sequential Dependence Assumption and bag of words assumption, respectively.

On using nDCG values obtained for Bi-Gram IR and VSM using two-tailed t-test, we get $p\_value = 9.81 \times 10^{-10}$. Since p_value is less than α, null hypothesis is rejected.

Therefore, we can say that *VSM performs better than Bi-Gram IR with respect to nDCG in Information Retrieval on Cranfield dataset under Sequential Dependence Assumption and bag of words assumption,respectively.*

Reasons as to why VSM performs better than bi-gram IR

- Based on the results, it seems that co-occurrences of words in this corpus is not very common. That may happen if documents are not closely related.
- Bigrams lead to more sparsity as compared to vector space model. This might result in poorer results.
- In presence of synonyms, VSM would perform better than nGram IR. To give an example:

- Consider a document with a single sentence - *Computer Architecture and Organization*. It can be decomposed in ["Computer", "Architecture", "Organization"] in VSM and ["ComputerArchitecture", "Architecture-Organization"] in bi-gram representation.
- If query is *Computer Design and Organization*, it will be decomposed into ["Computer", "Design", "Organization"] in VSM and ["Computer-Design", "DesignOrganization"] in bi-gram representation.
- Clearly, VSM would perform better in this case than bi-gram IR.

7.5 Limitations

– Polysemy - Since there is no word sense disambiguation being done polysemy still persists.
– nGram IR also fails to deal with synonymous words.
– Unless the value of n is large, there is no significant change in the number of dimensions.
– In all the above mentioned IR systems we don't try to capture the underlying conceptual similarity between documents or between documents and queries. Hence the results might not be accurate.

8 LSA

As we have seen many drawbacks still persist. To deal with the drawbacks we propose LSA. Advantages of LSA are as follows-

– Synonymy - done by capturing the latent (hidden) semantic relationships between words. While LSA doesn't explicitly identify synonyms, it can capture the underlying semantic similarity between words.
– Dimensionality - LSA reduces the dimensionality of the original term-document matrix using Singular Value Decomposition (SVD), creating a lower-dimensional space where the relationships between terms and documents are more densely represented.This reduction in dimensionality will give us accurate and reliable similarity measures.

8.1 Hyperparamter Tuning in LSA

As discussed above the original term-document matrix is approximated as a lower-rank matrix.By retaining only the top k singular values and their corresponding singular vectors, where k is typically much smaller than the original dimensions of the matrix, hence aiding in dimensionality reduction. We try to estimate the best value of k based on its performance of the system on the metric nDCG at rank 10, since it is the richest measure among all the measures and nDCG at rank 10 is chosen because as we've seen from the plots above that nDCG seems to stabilise as rank increases.

We obtain the plot below for varying k values like [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000] and the nDCG at rank 10 -

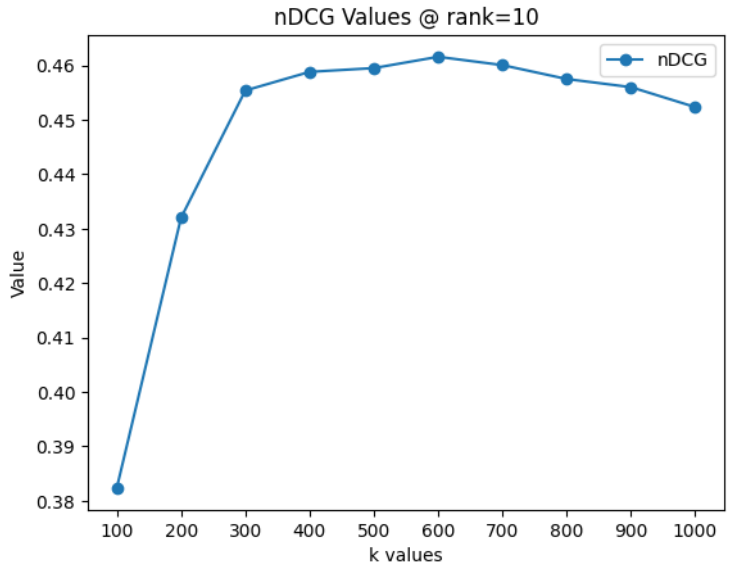

Fig. 5. nDCG values vs number of latent concepts(k) (LSA)

From the above plot we can take the optimal value for k to be 550.And this value is used for further analysis.

8.2 Time Taken

Time taken for LSA to run is 39.745627641677856 seconds

8.3 Results

k	Precision	Recall	$F_{0.5}$-score	MAP	nDCG
1	0.627	0.103	0.289	0.627	0.509
2	0.553	0.182	0.361	0.684	0.481
3	0.492	0.233	0.372	0.701	0.459
4	0.437	0.270	0.362	0.698	0.447
5	0.396	0.297	0.348	0.685	0.441
6	0.374	0.331	0.343	0.675	0.447
7	0.350	0.355	0.332	0.662	0.451
8	0.327	0.378	0.319	0.654	0.455
9	0.308	0.396	0.308	0.646	0.458
10	0.291	0.413	0.296	0.639	0.462

Table 5. Performance metrics for different values of k(LSA)

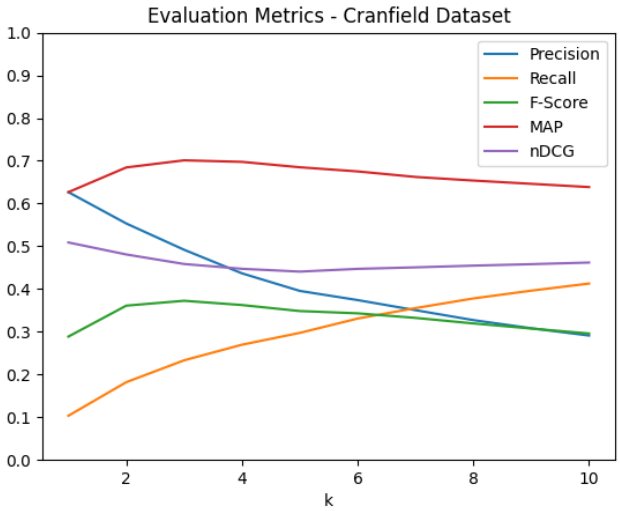

Fig. 6. Performance metrics plot for different values of k (LSA)

8.4 Hypothesis Testing for VSM and LSA

We compare LSA and VSM using hypothesis testing.

***null hypothesis*:** - LSA is not significantly better than VSM with respect to evaluation measure nDCG in Information Retrieval on Cranfield dataset under Bag of Words assumption

On using the nDCG values obtained for LSA and VSM using two tailed t-test, we get $p\_value = 1.27 \times 10^{-3}$. Since p_value is less than α, null hypothesis is rejected.

Therefore, we can say that *LSA performs better than VSM with respect to nDCG in Information Retrieval on Cranfield dataset under bag of words assumption.*

8.5 LSA with Query Expansion

It is established that LSA can deal with synonymy we intend to test this hypothesis, our intuition is that if LSA were to deal with synonymy then LSA + QE should have similar performance.

Time Taken The time for LSA + QE to run is - 45.42310953140259 seconds

k	Precision	Recall	$F_{0.5}$-score	MAP	nDCG
1	0.644	0.109	0.302	0.644	0.510
2	0.538	0.177	0.351	0.684	0.467
3	0.456	0.218	0.347	0.686	0.438
4	0.412	0.255	0.342	0.687	0.431
5	0.383	0.291	0.338	0.681	0.429
6	0.356	0.319	0.327	0.670	0.431
7	0.328	0.337	0.312	0.658	0.432
8	0.307	0.358	0.300	0.649	0.436
9	0.296	0.388	0.296	0.637	0.446
10	0.279	0.402	0.284	0.630	0.449

Table 6. Performance metrics for different values of k(LSA+QE)

Results Plot we obtained is as follows -

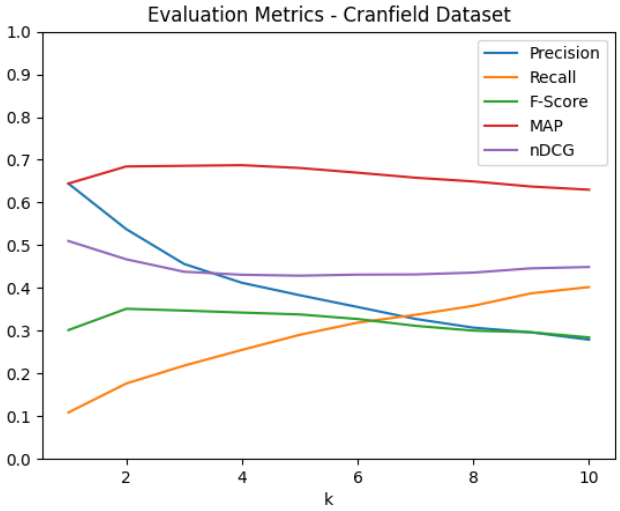

Fig. 7. Performance metrics plot for different values of k (LSA+QE)

8.6 Hypothesis Testing for LSA vs LSA + QE

We check whether for this dataset LSA + QE does perform any different than LSA alone using hypothesis testing.

null hypothesis: - LSA with Query Expansion is not significantly better than LSA with respect to nDCG in Information Retrieval on Cranfield Dataset under Bag of Words assumption

On using the nDCG values obtained for LSA and LSA with Query Expansion, using two tailed t-test, we get $p\_value = 4.84 \times 10^{-5}$. Since p value is less than α, null hypothesis is rejected.

Therefore, we can say that *LSA performs better than LSA with Query Expansion with respect to nDCG in Information Retrieval on Cranfield Dataset under bag of words assumption*

Reason as to why this might be-

- Similar arguments as in the case of VSM vs VSM+QE are valid here.
- It is important to note that these observations are subject to the dataset.

9 Normalized - LSA

In this approach we modify the entries of the term-document matrix used for SVD, in vanilla LSA the entries represent the term-frequencies(tf), in this approach we normalise these frequencies. By diving all the tf's for a document by the total number of tokens present in the document.

Intuition behind this is - better explained using an example, say we have Query 'what is a high-speed digital computer?' , say Doc1 contains the words 'digital','computer' with some frequencies but the total number of tokens is 500 vs say Doc2 contains the words 'digital','computer' with same frequencies but the total number of tokens is 50, then we would expect Doc2 to be more relevant than Doc1. This is captured by normalising the term frequencies.

9.1 Time Taken

Time taken to run Normalized-LSA is 41.16562867164612 seconds

9.2 Results

k	Precision	Recall	$F_{0.5}$-score	MAP	nDCG
1	0.684	0.116	0.320	0.684	0.553
2	0.593	0.193	0.387	0.736	0.524
3	0.518	0.248	0.394	0.743	0.494
4	0.471	0.293	0.391	0.739	0.486
5	0.435	0.329	0.383	0.730	0.483
6	0.396	0.356	0.365	0.720	0.482
7	0.368	0.377	0.350	0.710	0.482
8	0.342	0.396	0.335	0.701	0.485
9	0.323	0.416	0.323	0.694	0.489
10	0.305	0.433	0.310	0.681	0.492

Table 7. Performance metrics for different values of k(norm-LSA)

Plot is as below-

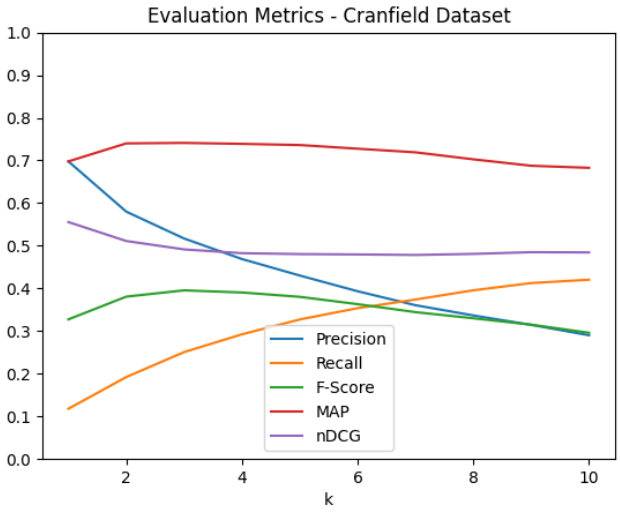

Fig. 8. Performance metrics plot for different values of k (norm-LSA)

9.3 Hypothesis Testing for LSA vs Normalised LSA

We check whether Normalised LSA performs better than LSA using hypothesis testing.

***null hypothesis*:** - Normalised LSA is not significantly better than LSA with respect to evaluation measure nDCG in Information Retireval on Cranfield Datset under Bag of Words assumption.

On using the nDCG values obtained for LSA and Normalised LSA, using two tailed t-test, we get $p\_value = 7.83 \times 10^{-9}$. Since p_value is less than α, null hypothesis is rejected. Therefore, we can say that *Normalised LSA performs better than LSA with Query Expansion with respect to nDCG in Information Retireval on Cranfield dataset under bag of words assumption.*

10 Limitations of above LSA retrieval methods

- Bag of words assumption - though we try to reduce documents to concepts we still ignore the sequential and contextual information of words within a document, potentially oversimplifying the semantic meaning.
- LSA can't handle polysemy though it might be successful to an extent by concept mining but largely it fails to do so.

11 LSA Bigram Hybrid

To combat the first limitation we use lsa hybrid technique with incorporates bigrams into the original term document matrix, terms are no longer single tokens but also bigrams. The entries of this matrix are either term-frequency or bigram frequency.Rest remains the same as LSA that is we use SVD for concept mining etc.

11.1 Time Taken

Time taken for LSA + bi-gram hybrid to run - 359.46420526504517 seconds

11.2 Results

k	Precision	Recall	$F_{0.5}$-score	MAP	nDCG
1	0.640	0.103	0.291	0.640	0.503
2	0.553	0.176	0.358	0.693	0.477
3	0.498	0.231	0.375	0.710	0.462
4	0.449	0.274	0.371	0.701	0.455
5	0.404	0.302	0.356	0.692	0.445
6	0.369	0.327	0.339	0.683	0.445
7	0.340	0.348	0.323	0.669	0.445
8	0.320	0.371	0.313	0.657	0.449
9	0.300	0.387	0.300	0.651	0.451
10	0.284	0.407	0.290	0.645	0.457

Table 8. Performance metrics for different values of k(LSA Hybrid)

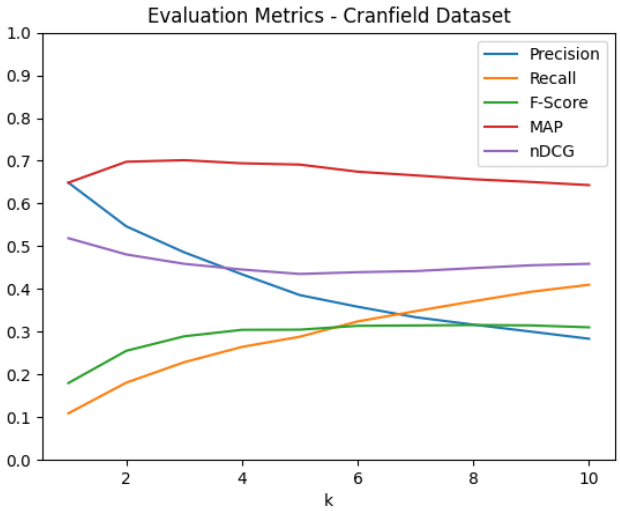

Fig. 9. Performance metrics plot for different values of k (LSA Hybrid)

11.3 Hypothesis Testing for LSA vs LSA - Hybrid

We check whether for this dataset, LSA-Hybrid performs any better than LSA using hypothesis testing.

***null hypothesis*:** - LSA-Hybrid is not significantly better than LSA with respect to evalutaion measure nDCG in Information Retrieval on Cranfield dataset.

On using nDCG values obtained for LSA-Hybrid and LSA, using two-tailed t-test, we get $p\_value = 0.23665$. Since p_value is greater than α, we can not reject null hypothesis.

Therefore, we can say that *LSA-Hybrid does not perform significantly better than LSA with respect to nDCG in Information Retireval on Cranfield dataset.*

Possible reasons behind similar performance of LSA and LSA-Hybrid

- This observation just re-enforces our believe of bigrams performing poorly.
- It strongly suggest that co-occurence of words is rare in this corpus and adding bigrams to individual tokens does not add much information to the IR system.
- This also strengthens the results obtained when comparing VSM with Bi-Gram IR.

12 Limitations still unaddressed

- All the above LSA models don't deal with polysemy.
- LSA isn't able enough to capture the nuances of language semantics due to its reliance on statistical patterns alone.
- LSA is highly sensitive to the number of latent semantic concepts.And is computationally expensive.
- There are approaches like ESA which uses external knowledge to better represent documents and achieve better results.