NLP Project Report

By Ranjith Kumar (BS17B024) and Sai N (BE17B010)

Indian Institute of Technology, Madras

Abstract. The previous search engine performed decently in the cranfield dataset, but there is always room for improvement. Here, we propose some methodologies to improve the retrieval performance (nDCG score) of our model.

Keywords: ESA · Inverted Index · LSA · Word2Vec · Query Expansion · nDCG score.

1 Background

1.1 Limitations of the previous search engine

– In the failed examples, the search engine performed poorly because the vector space model saw every document as a **bag of words**. Order of occurrence of words was not taken into account. And if the truly relevant document didn't have any common words with the query, then the model didn't retrieve the document as a relevant one.
– Other limitations for our vector space model were:
  – **Polysemy** may result in retrieval of irrelevant documents =¿ Loss in precision
  – **Synonymy** may result in failing to retrieve relevant documents =¿ Loss in recall
– And there were several words in queries and documents, which were preprocessed poorly, and these preprocessing errors in turn resulted in improper retrieval of relevant documents.

1.2 Methodologies proposed for improving the model

On seeing the above limitations of the previous model, we concluded that the following approaches will fetch better results.

– For making our model better than a simple "bag of words" model - Incorporate **word relatedness knowledge** into our model
– For more relevant retrieval - Include **Title** into the vector space representation of the document. We could even give some higher weight to the title than the body of the document while retrieving.
– For solving retrieval mistakes due to preprocessing errors - Make some **tweaks in the preprocessing stages** as well

1.3 Word Relatedness Knowledge

Our previous Vector Space Model neither had any knowledge on the meaning of a word nor treated synonyms/related words in a different way. It just treated each query/document as a bag of words. Hence, we tried to improve this situation by incorporating word relatedness knowledge into the model. All queries and documents in the cranfield dataset are aerospace related and a close examination on the failed examples showed us that, for improved performance, we needed to know word relatedness between technical words. Hence, we decided Explicit Semantic Analysis (ESA) would be an ideal approach to start with. Instead of using Wikipedia as the corpus, We used a set of aerospace related wiki-articles for better performance And then, later on, we tried implementing LSA and Word2Vec models as well.

1.4 Tweaks in the preprocessing stages

Upon looking at some failed examples, we found that some of the wrongly retrieved documents were due to poor preprocessing of query/documents. To state some examples -

– For the query - ”is there any information on how the addition of a /boat-tail/ affects the normal force on the body from various angles of incidence .”, the words “/boat-tail/” stay as “/boat-tail/” itself even after preprocessing.
– For the query - “papers on internal /slip flow/ heat transfer studies” - the words “/slip flow/” remain as “/slip” and “flow/” even after preprocessing and hence could not retrieve relevant documents. In the truly relevant documents, these words are present as “slip” or “flow” or “slip-flow”.
– And similar queries like these. This error is due to the weakness of the Penn Treebank tokenizer.

2 Implementing Tweaks in preprocessing

To solve this, we just targeted these symbols - “ / ”, “ ? ”, “ - ”, “ ’ ” (backslash, question mark, hyphen, apostrophe), marked these as word boundaries and stripped them off, which increased the quality of types in docs, queries, Wikipedia articles.

3 Implementing ESA

Steps

1. Building inverted index using Wikipedia articles
2. Query Expansion
3. Building document index and ranking documents for each query
4. Calculating Evaluation metrics

3.1 Building Wikipedia Inverted Index using Wikipedia Articles

For acquiring the relevant set of Wikipedia articles, firstly, we looked at the most repeated "technical" words in the cranfield documents & queries and extracted 23 words from it. And then gave those 23 words as inputs to Petscan and collected 31 relevant titles (All are titles of various Wikipedia articles). With those 312 titles, we compiled a dictionary (wiki_articles) with titles as keys and the content inside the corresponding articles as values.

After preprocessing queries, docs and content inside each of the Wikipedia articles, we had:

- 4684 types in docs
- 702 types in queries
- 13799 types in the content inside all of Wikipedia article.

The number of types in the wikipedia articles can be increased by extracting more articles. We couldn't do it at this time, because of the extended run-time. But, nevertheless can surely be considered for future works.

With the preprocessed types in articles, we build an inverted index (a dictionary - wiki_index) with types as keys and their TF-IDF vector in wikipedia concept space as values.We dumped this huge index into a json file (wiki_index.json), in order to use it again without running the extraction code again.

3.2 Query Expansion

For each word "(type)", the inverted index contains a TF-IDF vector in the wikipedia concept space. With the help of wiki_index, We then implemented query expansion for each query.

How? - Say that the input query has k words, for each of these words, we would calculate its cosine similarity with each of the wikipedia types (t) and append only the types which have cosine similarity more than the specified threshold. With this way, for k words in a query, we would have some additional similar words (query expansion). We would also assign the cosine similarity of the query word and the type as weights to the type.

3.3 Steps III and IV

After query expansion, we do the routine work which we did in the previous search engine (Build doc index, Rank the documents for each query and calculate evaluation metrics). The only difference here is that we have expanded queries with each word in the query having a weight, so while calculating doc index or ranking documents, we would take their respective weights into account.

4 Implementing LSA

Latent Semantic Analysis is an introspective method, where it computes word relatedness using the corpus itself. The underlying principle is that similar words will occur in similar pieces of text. Unlike ESA, the concepts in LSA are latent. Hence, we have represented the documents and queries in terms of these latent concepts space and then have found cosine similarity between them for relevance. We implemented this LSA variant and tuned some of its parameters such as number of topics to get a nDCG score of 0.51 at rank 10, which is by far the best we have achieved. One of the important parameters to decide here is the number of topics to use. We fixed the number of topics in gensim's LSI to be 250 after plotting the number of topics vs nDCG score obtained at rank - 10.

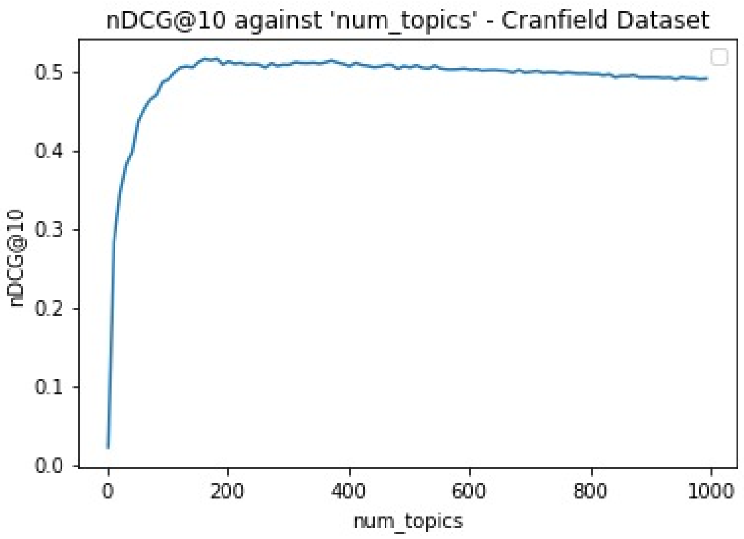


Fig. 1. Number of topics Vs nDCG score at rank-10

Note that Implementing LSA is the only step that varies here. Other than that, We have followed the same procedure (as discussed above) for other processes such as preprocessing tweaks, query expansion, building document index,ranking documents for each query and calculation evaluation metrics.

5 Word2Vec

The Word2Vec Skip-gram model, for example, takes in pairs (word1, word2) generated by moving a window across text data, and trains a 1-hidden-layer neural network based on the synthetic task of given an input word, giving us a predicted probability distribution of nearby words to the input. A virtual one-hot encoding of words goes through a 'projection layer' to the hidden layer; these projection weights are later interpreted as the word embeddings.

6 Comparing nDCG Scores

Model	nDCG @ $r1$	nDCG @ $r10$
Bag of words	0.5166	0.4611
ESA	0.4426	0.4646
LSA	0.57	0.5111
Word2Vec	0.5011	0.4583

7 Inferences and Conclusion

On comparing the nDCG scores, we get that the LSA model performs best for our cranfield model with a high nDCG score of 0.5111 at rank-10. All the other three model's performance scores are somewhat similar. Hence, we can deploy LSA model whenever necessary.

8 Future Works

1. A larger set of wikipedia articles (maybe even the entire dump) can be used for ESA
2. In ESA, we assumed that each article is independent of the other. We could try exploring some other variant of ESA, where this assumption would be relaxed.

9 References

1. Class notes and Lectures
2. ESA, LSA research papers
3. Medium articles on LSA implementation
4. Word2Vec on TensorFlow tutorials

10 Relevant Plots

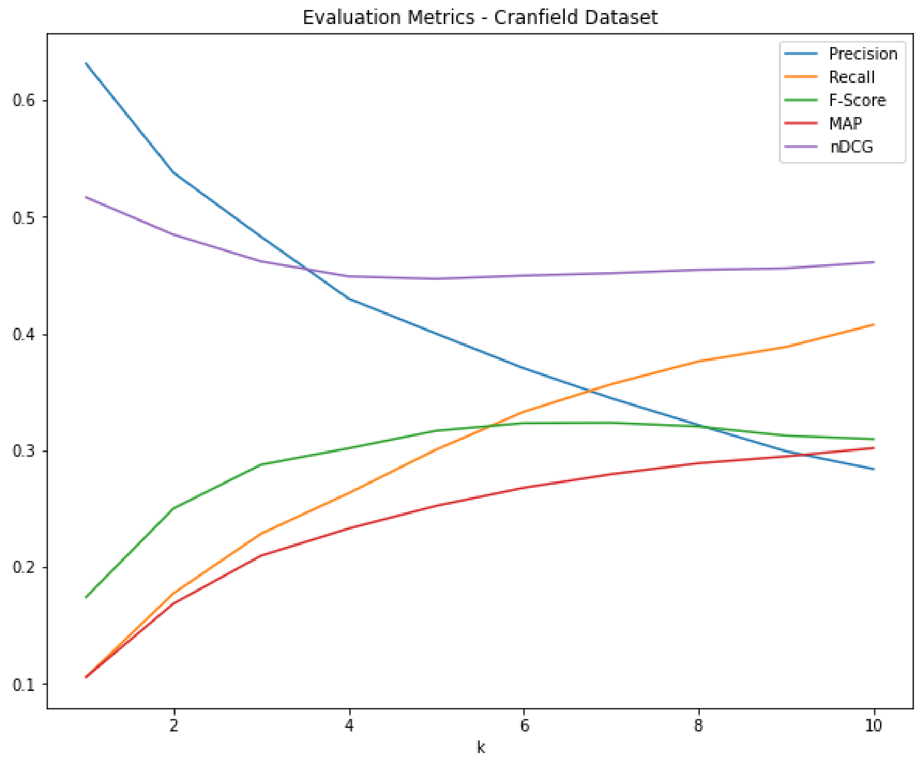
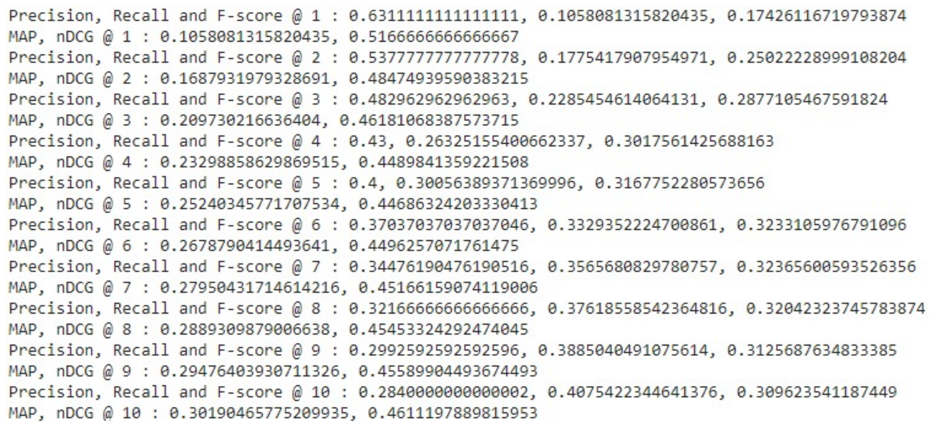


Fig. 2. Evaluation plot - Previous model.



```
Precision, Recall and F-score @ 1 : 0.6311111111111111, 0.1058081315820435, 0.17426116719793874
MAP, nDCG @ 1 : 0.1058081315820435, 0.5166666666666667
Precision, Recall and F-score @ 2 : 0.5377777777777778, 0.1775417907954971, 0.25022228999108204
MAP, nDCG @ 2 : 0.1687931979328691, 0.48474939590383215
Precision, Recall and F-score @ 3 : 0.482962962962963, 0.2285454614064131, 0.2877105467591824
MAP, nDCG @ 3 : 0.209730216636404, 0.46181068387573715
Precision, Recall and F-score @ 4 : 0.43, 0.2632515540066237, 0.3017561425688163
MAP, nDCG @ 4 : 0.23298858629869515, 0.4489841359221508
Precision, Recall and F-score @ 5 : 0.4, 0.30056389371369996, 0.3167752280573656
MAP, nDCG @ 5 : 0.25240345771707534, 0.44686324203330413
Precision, Recall and F-score @ 6 : 0.37037037037037046, 0.3329352224700861, 0.3233105976791096
MAP, nDCG @ 6 : 0.2678790414493641, 0.4496257071761475
Precision, Recall and F-score @ 7 : 0.34476190476190516, 0.3565680829780757, 0.32365600593526356
MAP, nDCG @ 7 : 0.27950431714614216, 0.45166159074119006
Precision, Recall and F-score @ 8 : 0.32166666666666666, 0.37618558542364816, 0.32042323745783874
MAP, nDCG @ 8 : 0.2889309879006638, 0.45453324292474045
Precision, Recall and F-score @ 9 : 0.2992592592592596, 0.3885040491075614, 0.3125687634833385
MAP, nDCG @ 9 : 0.29476403930711326, 0.45589904493674493
Precision, Recall and F-score @ 10 : 0.2840000000000002, 0.4075422344641376, 0.309623541187449
MAP, nDCG @ 10 : 0.30190465775209935, 0.4611197889815953
```

Fig. 3. Evaluation measures - Previous model.

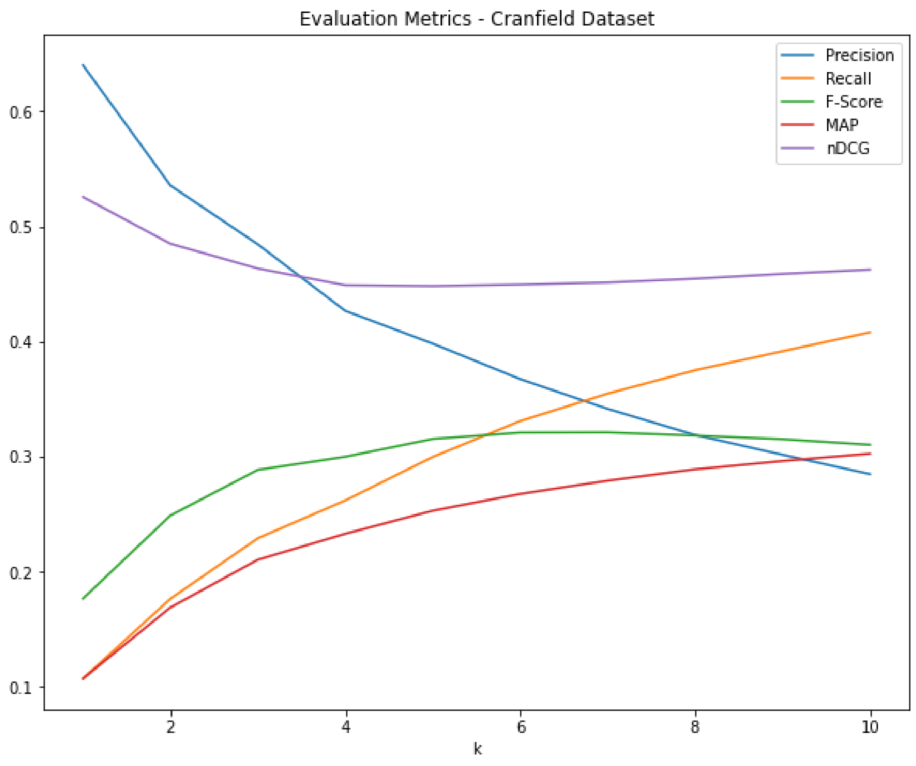
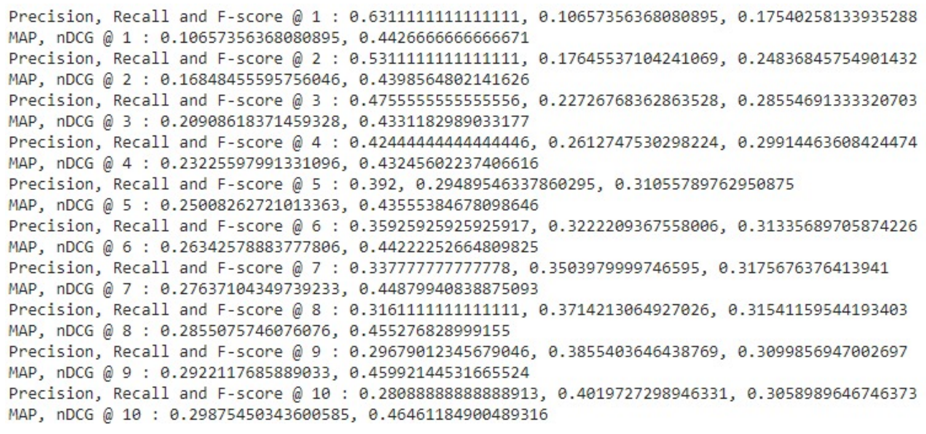


Fig. 4. Evaluation plot - ESA model.



```
Precision, Recall and F-score @ 1 : 0.6311111111111111, 0.10657356368080895, 0.17540258133935288
MAP, nDCG @ 1 : 0.10657356368080895, 0.4426666666666671
Precision, Recall and F-score @ 2 : 0.5311111111111111, 0.17645537104241069, 0.24836845754901432
MAP, nDCG @ 2 : 0.16848455595756046, 0.4398564802141626
Precision, Recall and F-score @ 3 : 0.4755555555555556, 0.22726768362863528, 0.28554691333320703
MAP, nDCG @ 3 : 0.20908618371459328, 0.4331182989033177
Precision, Recall and F-score @ 4 : 0.42444444444444446, 0.2612747530298224, 0.29914463608424474
MAP, nDCG @ 4 : 0.23225597991331096, 0.43245602237406616
Precision, Recall and F-score @ 5 : 0.392, 0.29489546337860295, 0.31055789762950875
MAP, nDCG @ 5 : 0.25008262721013363, 0.43555384678098646
Precision, Recall and F-score @ 6 : 0.35925925925925917, 0.3222209367558006, 0.31335689705874226
MAP, nDCG @ 6 : 0.26342578883777806, 0.44222252664809825
Precision, Recall and F-score @ 7 : 0.3377777777777778, 0.3503979999746595, 0.3175676376413941
MAP, nDCG @ 7 : 0.27637104349739233, 0.44879940838875093
Precision, Recall and F-score @ 8 : 0.3161111111111111, 0.3714213064927026, 0.31541159544193403
MAP, nDCG @ 8 : 0.2855075746076076, 0.455276828999155
Precision, Recall and F-score @ 9 : 0.29679012345679046, 0.3855403646438769, 0.3099856947002697
MAP, nDCG @ 9 : 0.2922117685889033, 0.45992144531665524
Precision, Recall and F-score @ 10 : 0.28088888888888913, 0.4019727298946331, 0.3058989646746373
MAP, nDCG @ 10 : 0.29875450343600585, 0.46461184900489316
```

Fig. 5. Evaluation measures - ESA model.

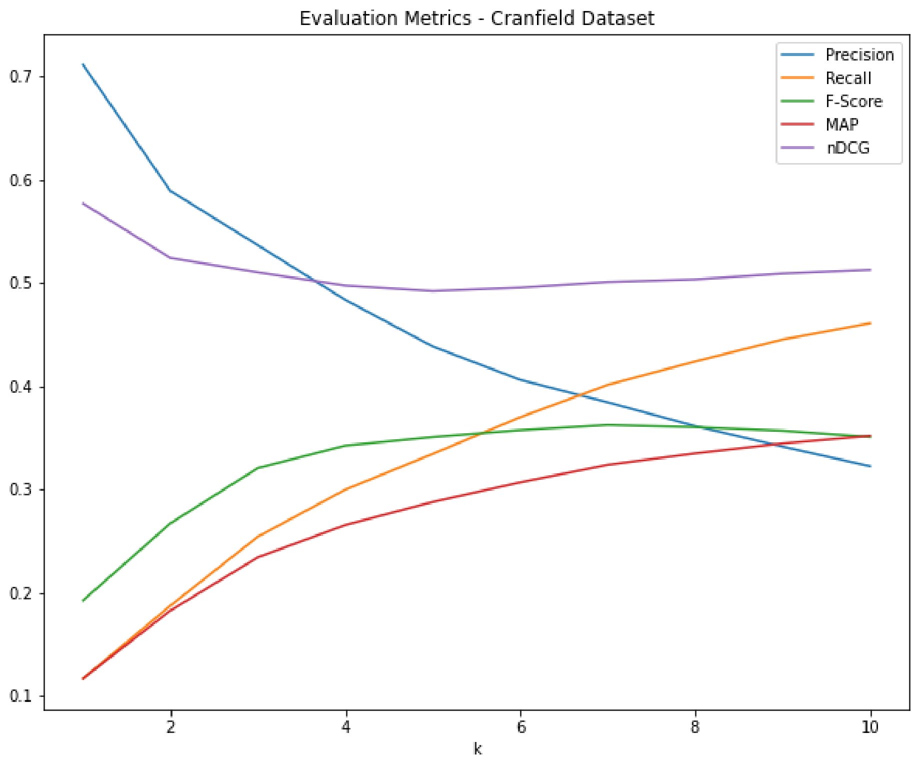
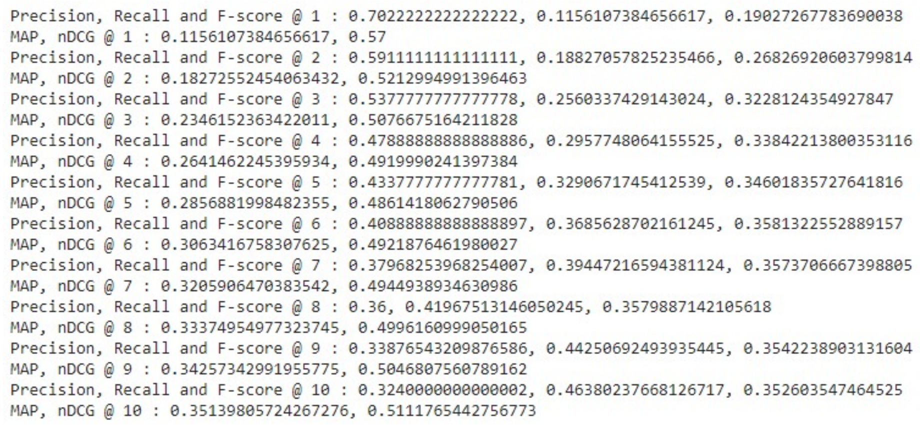


Fig. 6. Evaluation plot - LSA.



```
Precision, Recall and F-score @ 1 : 0.7022222222222222, 0.1156107384656617, 0.19027267783690038
MAP, nDCG @ 1 : 0.1156107384656617, 0.57
Precision, Recall and F-score @ 2 : 0.5911111111111111, 0.18827057825235466, 0.26826920603799814
MAP, nDCG @ 2 : 0.18272552454063432, 0.5212994991396463
Precision, Recall and F-score @ 3 : 0.5377777777777778, 0.2560337429143024, 0.3228124354927847
MAP, nDCG @ 3 : 0.2346152363422011, 0.5076675164211828
Precision, Recall and F-score @ 4 : 0.47888888888888886, 0.2957748064155525, 0.33842213800353116
MAP, nDCG @ 4 : 0.2641462245395934, 0.4919990241397384
Precision, Recall and F-score @ 5 : 0.4337777777777781, 0.3290671745412539, 0.34601835727641816
MAP, nDCG @ 5 : 0.2856881998482355, 0.4861418062790506
Precision, Recall and F-score @ 6 : 0.40888888888888897, 0.3685628702161245, 0.3581322552889157
MAP, nDCG @ 6 : 0.3063416758307625, 0.4921876461980027
Precision, Recall and F-score @ 7 : 0.37968253968254007, 0.39447216594381124, 0.357370667398805
MAP, nDCG @ 7 : 0.3205906470383542, 0.4944938934630986
Precision, Recall and F-score @ 8 : 0.36, 0.41967513146050245, 0.3579887142105618
MAP, nDCG @ 8 : 0.33374954977323745, 0.4996160999050165
Precision, Recall and F-score @ 9 : 0.33876543209876586, 0.44250692493935445, 0.3542238903131604
MAP, nDCG @ 9 : 0.34257342991955775, 0.5046807560789162
Precision, Recall and F-score @ 10 : 0.3240000000000002, 0.46380237668126717, 0.352603547464525
MAP, nDCG @ 10 : 0.35139805724267276, 0.5111765442756773
```

Fig. 7. Evaluation measures - LSA.

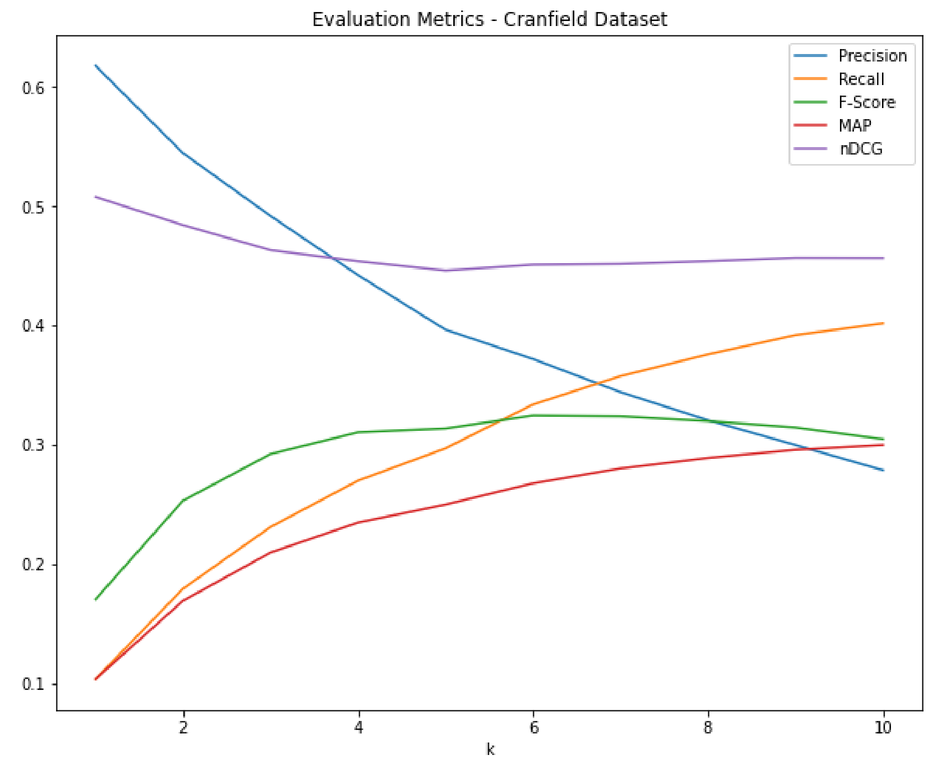
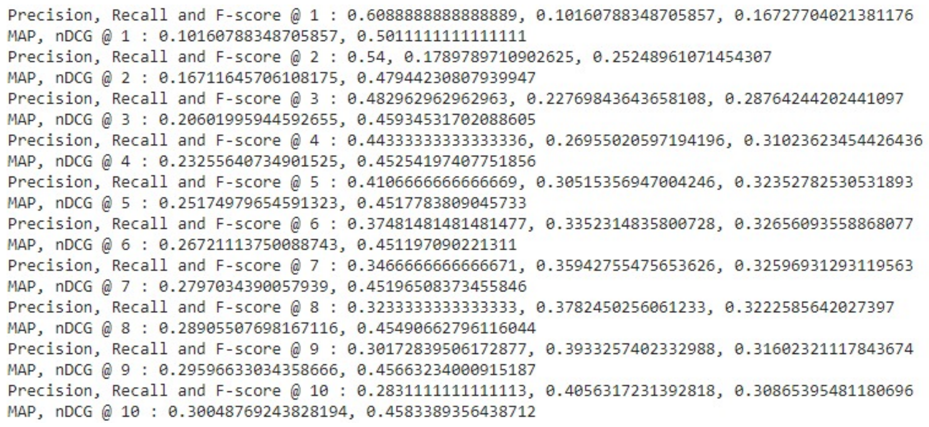


Fig. 8. Evaluation plot - Word2Vec.



```
Precision, Recall and F-score @ 1 : 0.6088888888888889, 0.10160788348705857, 0.16727704021381176
MAP, nDCG @ 1 : 0.10160788348705857, 0.5011111111111111
Precision, Recall and F-score @ 2 : 0.54, 0.1789789710902625, 0.25248961071454307
MAP, nDCG @ 2 : 0.16711645706108175, 0.47944230807939947
Precision, Recall and F-score @ 3 : 0.482962962962963, 0.22769843643658108, 0.28764244202441097
MAP, nDCG @ 3 : 0.20601995944592655, 0.45934531702088605
Precision, Recall and F-score @ 4 : 0.44333333333333336, 0.26955020597194196, 0.31023623454426436
MAP, nDCG @ 4 : 0.23255640734901525, 0.45254197407751856
Precision, Recall and F-score @ 5 : 0.4106666666666669, 0.30515356947004246, 0.32352782530531893
MAP, nDCG @ 5 : 0.25174979654591323, 0.4517783809045733
Precision, Recall and F-score @ 6 : 0.37481481481481477, 0.3352314835800728, 0.32656093558868077
MAP, nDCG @ 6 : 0.26721113750088743, 0.451197090221311
Precision, Recall and F-score @ 7 : 0.3466666666666671, 0.35942755475653626, 0.32596931293119563
MAP, nDCG @ 7 : 0.2797034390057939, 0.45196508373455846
Precision, Recall and F-score @ 8 : 0.3233333333333333, 0.3782450256061233, 0.3222585642027397
MAP, nDCG @ 8 : 0.28905507698167116, 0.45490662796116044
Precision, Recall and F-score @ 9 : 0.30172839506172877, 0.3933257402332988, 0.31602321117843674
MAP, nDCG @ 9 : 0.29596633034358666, 0.45663234000915187
Precision, Recall and F-score @ 10 : 0.2831111111111113, 0.4056317231392818, 0.30865395481180696
MAP, nDCG @ 10 : 0.30048769243828194, 0.4583389356438712
```

Fig. 9. Evaluation measures - Word2Vec.