# Information Retrieval

Abhiram Tarimala and Abhranil Chakrabarti

Indian Institute of Technology, Madras
`ch17b033@smail.iitm.ac.in` , `ch17b034@smail.iitm.ac.in`

**Abstract.** The aim of this work is to improve the vector space model for information retrieval on the Cranfield dataset. The vector space model is kept as the baseline and the results are compared with Latent Semantic Indexing, Explicit Semantic Analysis, WordNet Query expansion, Spell Corrected Documents and Word2Vec and GloVe word embeddings.

**Keywords:** Information Retrieval · Vector Space Model · Latent Semantic Indexing · Explicit Semantic Analysis · Word2Vec

## 1 Introduction

Information retrieval is the is the science of searching for information in documents, searching for documents themselves, searching for meta-data which describe documents or searching within databases, whether relational stand-alone databases or hyper textually-networked databases such as World Wide Web. An Information Retrieval (IR) model selects and ranks the document that is required by the user or the user has asked for in the form of a query
In today's world, information retrieval is used in almost every application from searching for songs and movies to finding a book in a library catalogue or searching for something or someone on the internet
An information retrieval system consists of the following components:

- Document Collection – the collection of documents over which any search is performed
- Set of Queries – the queries on which the search is performed for retrieval;
- Indexing System – searching and indexing methods to retrieve an ordered list of documents for every query;
- Evaluation Criteria – specified measures by which each system is evaluated, for example 'precision' and 'recall' as measures of relevance

## 2 Problem Definition

The specific problem that was worked on during the making of this report was Information Retrieval of documents from the Cranfield Dataset using two types of queries:

- **Cranfield Queries:** A set of queries available within the dataset along with the correct relevance of documents pertaining to the queries. This is used to test our method of Information Retrieval on metrics such as precision, recall, F-Score, Mean Average Precision and nDCG.
- **Custom Queries:** Queries that are created and entered by users at random. We seek to display the document IDs of the top 5 documents by relevance for a custom query.

While basic vector space models have been used to retrieve cranfield documents, there is scope for improvement in terms of the metrics mentioned above and in the speed of execution of the model.
The improvements that are sought to be made are described in Section 3, followed by a detailed explanation of each method in Section 5 and the way it was executed in Section 6. Section 7 highlights the results and a hypothesis test for each method against the baseline, the vector space model.

## 3 Motivation

The main motivation behind the solution was to address the limitations of the vector space model for information retrieval. The major limitations of the vector space model are :

- **Word Relatedness and Synonyms**: The Vector Space Model treats all the words as orthogonal to each other and hence doesn't account for the similarity between words and misses out on relations like synonymy, hyponymy, hypernymy, etc.
- **Order of Words**: The order in which the terms appear in the document is lost in the vector space representation
- **High Complexity**: With each word representing a dimension, a very high dimensional vector space is created. This leads to a high time complexity
- **Execution Speed**: Another issue we seek to address is making the vector space model run faster.

The above limitations can be overcome by employing methods to extract topics/features of documents and queries using the words present in them which would help in overcoming the limitation of words considered as independent in Vector space model. Also these models are efficient in computation and does not involve very high dimensions

## 4 Background and Related Work

**Latent Semantic Analysis**
Scott Deerwester introduced a method to index by taking advantage of implicit higher-order structure in the association of terms with documents and dubbed the method Latent Semantic Analysis (refer [1] in references). The technique used to do so is singular-value decomposition (SVD), in which a large term by

document matrix is decomposed into 3 components. Queries are represented as pseudo-document vectors formed from weighted combinations of terms and documents.

**Explicit Semantic Analysis**
Explicit Semantic Analysis was introduced by Evgeniy Gabrilovich and Shaul Markovitch (refer [2] in references) where they used Wikipedia articles as explicit concepts in which words and in turn documents, which are nothing but a collection of words, are represented in a vector space of wikipedia articles.

## 5 Proposed Methodology

### 5.1 Pre-Processing

The documents are first preprocessed to clean the data and convert it in a format more suitable for the information retrieval task. The techniques employed for pre-processing are as follows:

- **Conversion to lowercase**: All characters are converted to lowercase to ensure that tokens aren't treated differently due to case sensitivity
- **Segmentation** - Sentences are segmented using one of the two methods :
  - Naive segmentation : The sentences are segmented naively using the end of sentence characters like '.','?','!'
  - Punkt segmentation : The sentences are segmented using the PunktSentenceTokenizer which uses an unsupervised learning algorithm to build a model to identify abbreviation words, collocations, and words that start sentences to divide the text into a list of sentences. Unlike the simple top-down approach, which uses a set of punctuation marks determined from grammar, the Punkt Tokenizer mines its own rules from data
- **Spell Check** - A spell check is performed on the custom queries to correct any spelling errors which may be present. This is accomplished by generating a candidate set of words at an edit distance of upto 2 for every non-word in the query.
- **Tokenization** - Sentences are split into their constituent tokens using one of the two methods :
  - Naive Tokenization : Sentences are naively split based on whitespace and commas.
  - PennTreeBank Tokenization : Sentences are tokenized using the TreebankWordTokenizer which uses a Top-down approach to tokenize the text. Existing knowledge is used to create rules which are used for the purpose of tokenization
- **Inflection Reduction** : It is used to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form
  - Lemmatization: Lemmatization is the process of arriving at the lemma using a thorough and accurate morphological analysis of the words.

  - Stemming: Stemming makes uses of heuristic approaches such as the removal of common prefixes and suffixes and other such techniques to arrive at the reduced form.
- **StopWord Removal** : It is used to remove the words which does not add much meaning to a sentence. The list of such words is obtained from collection of stopwords present in the natural language toolkit corpus.

### 5.2 Techniques

**Latent Semantic Indexing** Using Singular Value Decomposition on the term-document matrix we created earlier, we can map our words and documents onto k latent concepts. This serves two purposes:

- Relations between words: Words are no longer treated as independent entities and relations between them are captured in the concepts
- Reduce the number of parameters: Instead of w*d parameters, we now store only k*(w+d) parameters.
  Where:
  w - number of words
  d - number of documents
  k - number of concepts taken

The query is then transformed to the concept space to compute the query-document similarity required for retrieval

**Explicit Semantic Analysis** We seek to use background knowledge, such as Wikipedia articles, as concepts and map words on this new concept space using TF-IDF scores. Further, documents and queries are mapped onto this space as a linear combination of the word vectors obtained. Cosine similarity is then used to find the similarity between queries and documents in this new space.

**Query Expansion using WordNet** Query expansion is needed to ensure that even if the query has synonyms or related terms to words present in the documents, it plays a role in retrieval.
WordNet is a large lexical English database. It has a collection of all words, which are grouped into sets of cognitive synonyms called synsets, each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations.
We propose using Wordnet to capture relations between words and use these relations to expand the queries.
Information is retrieved using these expanded queries using two methods:

- For each word in the document, check it's score with the score given to it after query expansion. Sum up the attained score for each word in the document for all documents and rank them based on the sum
- Ranking the documents by their cosine similarity with the expanded query

**Word Embeddings** The goal is to create vector representation of words which can capture similarity. One way of capturing this could be such that words with similar meaning have a higher cosine similarity. Some approaches of creating such representations are Word2Vec and GloVe.
Word2Vec can learn word representations in two ways:

– Skip-Gram: Neural network weights trained by trying to predict context words surrounding a target word (works better on larger datasets)
– Continuous Bag of Words: Neural network weights trained by trying to predict a target word from surrounding context words

The learnt representation can then be used to represent the document as:

$$D = \frac{1}{|D|}\sum_i \frac{w_i}{||w_i||}, \text{ where } w_i \text{ is the i-th word in the document}$$

The query-document similarity can be calculated as:

$$Similarity = \frac{1}{|Q|}\sum_i \frac{q_i^T D}{||q_i||||D||}, \text{ where } q_i \text{ is the i-th word in the document}$$

## 5.3 Evaluation

**Metrics** Many metrics are available for evaluating the information retrieval system such as mean Precision, mean Recall, MAP, mean F-score and mean nDCG.
The main metrics which we focused on are:

– **mean nDCG**: nDCG is an important metric since it can take the document relevance scores, which are available for the Cranfield dataset, into consideration
– **mean F-Score**: F-score gives importance to both precision and recall while computing the score and is thus key metric as we would ideally want both precision and recall to improve and not one at the cost of the other
– **mean Average Precision**: MAP takes the order of retrieved documents too into consideration due to which it is another key metric for the evaluation of the IR system

## 5.4 Hypothesis Testing:

The hypotheses we validate is that each of the algorithms we attempt, denoted by $A_1$ performs better than our **baseline - the simple Vector Space Model**, represented by $A_2$ on the **task of document retrieval** i.e, the task of retrieval of the top documents for each query, **using MAP and nDCG as the evaluation metrics** on the **Cranfield Dataset**, using the queries in the dataset.

We seek to perform a hypothesis test to ensure that the chosen model is truly better than the baseline and the difference in the compared metric is not due to statistical chance.

Instead of comparing the models using a single value of the chosen metric [Refer section 5.3], we split the queries into a test component and a train component and **create n samples of k queries each**.

By the Central Limit Theorem, taking sufficiently large random samples from the population with replacement , then the distribution of the sample means will be approximately normally distributed.

The number of queries in each cohort (k) and the number of cohorts (n) is taken as follows for each algorithm:

- **LSA**: k = 25 ; n = 100
- **ESA**: k = 25 ; n = 50
- **Query Expansion**: k = 10 ; n = 25

Since LSA executes fast, we could include the computation of more queries. We go on decreasing n,k for ESA and then WordNet since these take longer to execute and it isn't feasible to run so many queries for the algorithms with the limited computing power we possess.

Both the algorithms are run on all the created samples and the average metric for each of the n samples created is calculated. We then performed a one-tailed paired t-test on these observed metric to test our hypothesis with the value of alpha as 0.05, meaning we seek to say with 95% confidence that the new method outperforms the baseline

- Null Hypothesis: The null hypothesis (H0) assumes that the true mean difference ($\mu_{A1}$ - $\mu_{A2}$) - i.e., the mean of the metric of all the n samples is equal to zero
- Alternate Hypothesis: The one tailed alternative hypothesis (H1) assumes that ($\mu_{A1}$ - $\mu_{A2}$) is greater than zero.

### 5.5 Speedin Up

: The major bottleneck in the running of the vector space model is that it calculates the TF-IDF matrix of the documents each time it runs.
Even though the inverted index is calculated only at the beginning of the run, the calculation of the TF-IDF matrix from the inverted index is computationally expensive.
To prevent this, the TF-IDF values have been stored as a csv file and they are called at the beginning of the execution of the model. This ensures that all the required calculations are done before hand, saving execution time.

## 6 Experiments

### 6.1 Latent Semantic Indexing

First, the term-document tf-idf matrix is created. This is then decomposed using singular value decomposition to obtain the term-topic, document-topic matrix and the singular values corresponding to each topic. To reduce the topic space, the optimal number of topics had to be chosen. This was done with the help of topic coherence. It is a measure widely used to evaluate topic models. It uses the latent variables generated. Each generated topic has a collection of words and the average pairwise word similarity scores of the words in every topic is found. This was accomplished using the gensim library. The coherence values were plotted varying the number of topics from 100 to 1400 separated by intervals of 100. The plot is as given below. Beyond 1000, there was no significant

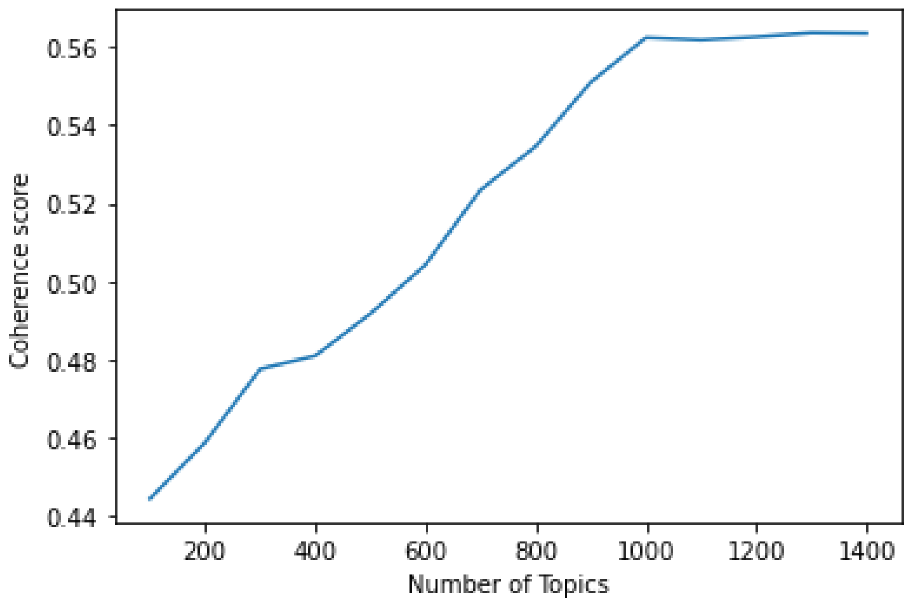


**Fig. 1.** Coherence scores with number of topics ranging from 100 to 1400 separated by intervals of 100

increase observed in the coherence scores. A final value of 700 was chosen for the number of topic based on the coherence score and the MAP and nDCG values obtained as a result.
Once the decomposition is done, each query is transformed into the latent space to compute the query-document similarity and arrange the documents in decreasing order of similarity scores.

### 6.2 Explicit Semantic Analysis

**Building the Wikipedia article vector space:** Currently, the English Wikipedia has over 6,308,927 articles, consuming upto 19gb+ of memory. Using such an expansive set of articles for ESA is neither necessary nor feasible, given the limited computing power.
To avoid processing large amounts of Wikipedia data, we proceed to select only relevant articles to the cranfield dataset. This was done as follows:

- A list was compiled of all the unique words in the cranfield dataset. The documents and queries combined have 8308 unique words.
- For each word, the top wikipedia article for that word search was retrieved using the **wikipedia** package in python and stored in a list.
- All the unique documents were retrieved from this list. This condensed dataset consists of 4543 articles with a vocabulary of 309230 unique words.
- A TF-IDF matrix was then constructed of all the words across each of the 4543 articles in wikipedia.
- This was then reduced by considering only the words that appear in the cranfield dataset to save space.
- This represents the **concept space** where each wikipedia article is a concept.
- Since getting the wikipedia articles and creating a TF-IDF matrix is a computationally expensive task, this was done once and the results were saved as a csv which is imported when running.

**Representing documents on Wikipedia vector space:** Once the wikipedia vector space was created, the nect task was to represent documents on this space. This was done as follows:

- A TF-IDF matrix was created among all the cranfield documents.
- For each word in a document, the representation of that word was taken in the wikipedia article space and that vector was multiplied with the TF-IDF value of the word corresponding to said document from the document matrix.
- The document's representation then becomes the sum of all the vectors of it's constituent words in the wikipedia space weighted with the word's TF-IDF value corresponding to that document.
- Using this methadology, a new matrix was created represting documents in the wikipedia article space - or the concept space. The dimensions of this are 1400x4503.

**Information Retrieval:**

- Given a query, the first task is to represent it on the concept space (wikipedia article space). This is done similar to how it's done for documents.

- A TF-IDF vector is first constructed for each query, using IDF values from the document matrix. Post that, every word's representative vector in the concept space is weighted with this TF-IDF value. All the word vectors of a query are added to give the representation of a query on the concept space of dimensions 1x4503.
- The cosine distance is found between the query and each document. This boils down to finding the cosine distances between two 1x4503 vectors 1400 times.
- The documents are arranged in the descending order of the cosine distance with the query.

## 6.3 Query Expansion with WordNet combined with Vector Space

**Query Expansion:** Multiple methods were used to expand the query using WordNet.

### Method 1

- First, a vocabulary was compiled by finding all the unique words in the documents and queries.
- The synsets were found for every word in the vocabulary.
- For each word in a given query, it's synsets were found. Similarity was computed between the synsets of the query word and sysnets of every other word in the vocabulary using the **Wu-Palmer Distance**.
  Wu-Palmer Distance $= 2 * \frac{depth(lcs(s_1,s_2))}{depth(s_1)+depth(s_2)}$
  The highest value of similarity between synsets is considered as the similarity between the words. Doing this for all the words in the query, we obtain a VxN matrix where V is the size of the vocabulary (8308) and N is the unique words in a particular query. The columns of this matrix are summed to give a Vx1 matrix, which represents the expanded query.

### Method 2

- For every word in the query, all of it's corresponding synsets are found from wordnet.
- For each synset of a word, it's different synonyms are found and each of them is appended to the query with a weight of 0.9*term frequency of the word.
- The newly constructed expanded query is passed as the query.

**Information Retrieval from Expanded Query:** Two methods are used to retrive documents from the expanded query.

### Method 1 - Cosine Similarity

- Cosine similarity is found between the expanded query an each of the 1400 documents. The documents are then ordered in decreasing order of the cosine similarity, similar to how it was done for the vector space model.

**Method 2 - Method of Aggregation**

- For every word in a document, it's score is assigned as the value of the word in the expanded query matrix. The score of a document is calculated by adding the score of each its words. This is done for all documents and they are ranked based on this score in descending order.

After trying all 4 combinations (2 methods in query expansion and 2 methods in information retrival) we observe that Method 2 in Query Expansion coupled with Method 1 in infotmation retrieval provides the best results and that is the method we chose to report in results.

### 6.4 Word Embeddings

Two approaches were used for word embeddings:

- A skip-gram Word2Vec trained on the Cranfield dataset. Some hyperparameter tuning was done to obtain a reasonable set of hyperparameters. The vector size, which is the dimensionality of each word vector was chosen to be 100. The window, which is the maximum distance between the target and surrounding words considered, was kept at 5. And the minimum count of words needed for it to be considered was 5.
- A pre-trained Glove model trained on an English Wikipedia dump was used. The chosen model was 'glove-wiki-gigaword-200' which owing to the larger dataset yielded better results than the model trained on the cranfield dataset

To create the document vector, each word in the document was linearly combined with weights equal to its tf-idf value in the document. The query vectors were also similarly created. Finally, the cosine similarity between the document and query vectors was used to arrange the documents in the retrieval order.

### 6.5 Spelling Correction

Spelling Correction was performed using Python's pyspellchecker library. First, non-words were identified and a candidate set was generated for only these words using a Levenshtein Distance algorithm by considering permutations at an edit distance of upto 2. The most probable word is then chosen from the set of candidates by selecting the words which are found more often in the package's frequency list.
The spell check was first performed on both the documents and the queries. However, there weren't any significant spelling errors in the existing documents and queries due to which the results too did not show any improvements.
It was then decided to include the spelling correction only for the custom queries since there is always a possibility of spelling errors for such queries.

## 7 Results

### 7.1 Hypothesis Testing

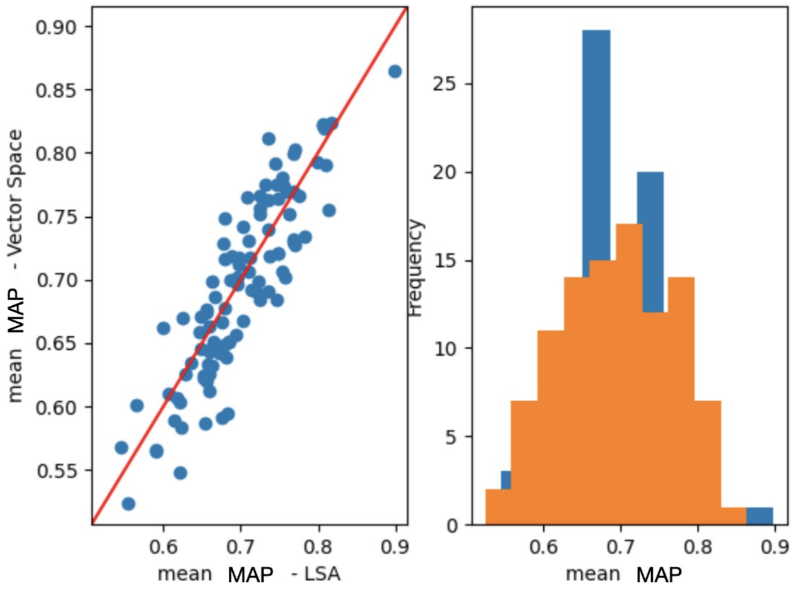


**Fig. 2.** The MAP values for LSA vs Vector Space - Orange is LSA, Blue is Vector Space

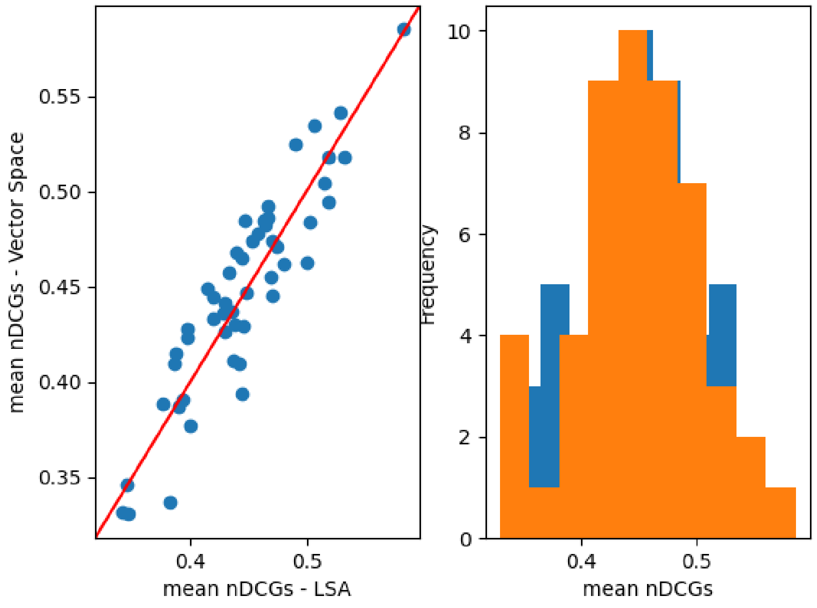


**Fig. 3.** The nDCG values for LSA vs Vector Space - Orange is LSA, Blue is Vector Space

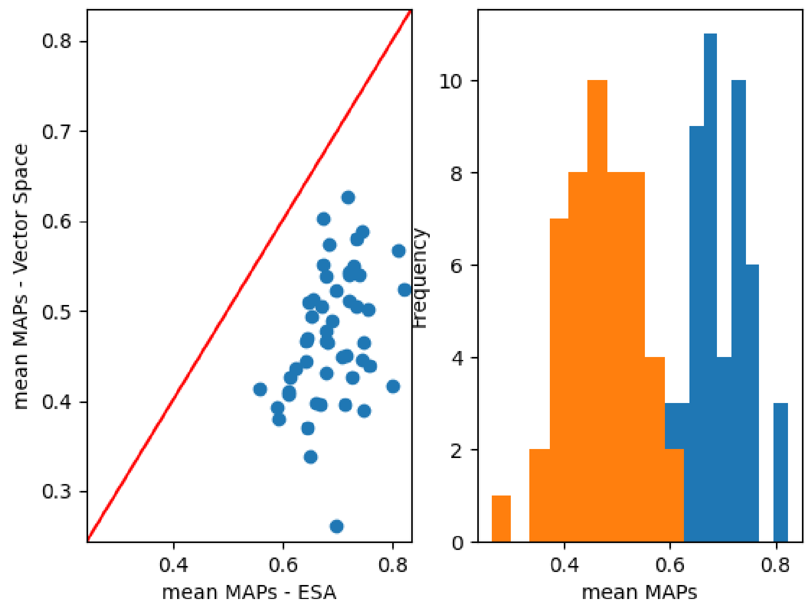


**Fig. 4.** The MAP values for ESA vs Vector Space - Orange is ESA, Blue is Vector Space

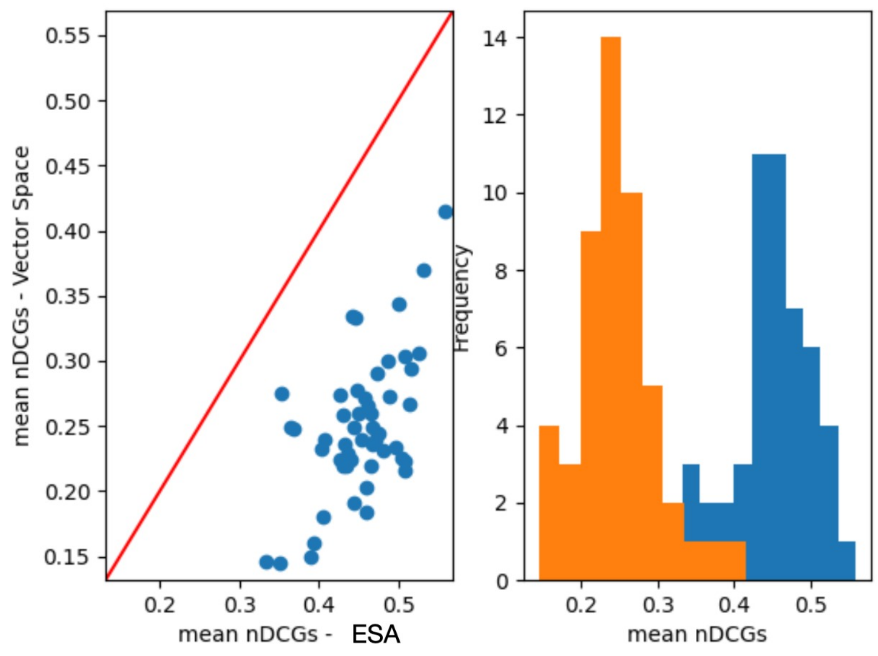


**Fig. 5.** The nDCG values for ESA vs Vector Space - Orange is ESA, Blue is Vector Space

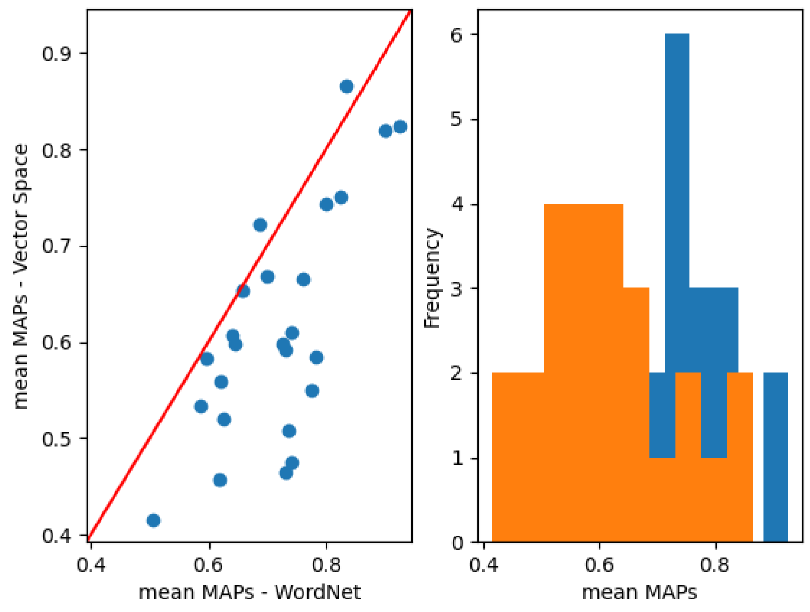


**Fig. 6.** The MAP values for WordNet vs Vector Space - Orange is WordNet, Blue is Vector Space

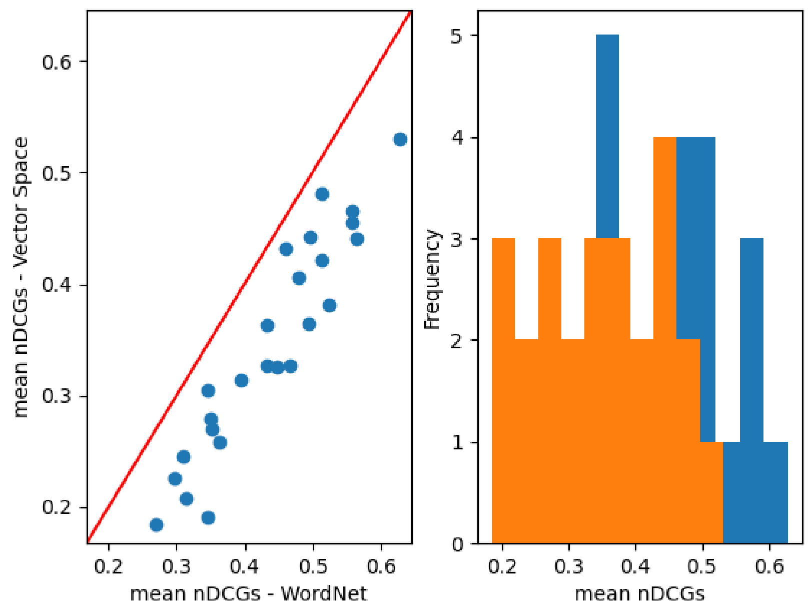


**Fig. 7.** The nDCG values for WordNet vs Vector Space - Orange is WordNet, Blue is Vector Space

## 7.2 Results Summary

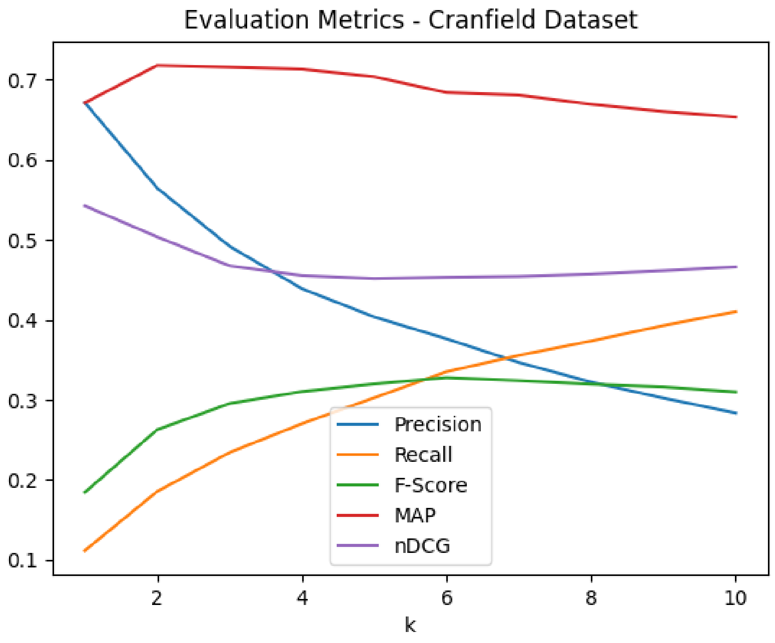
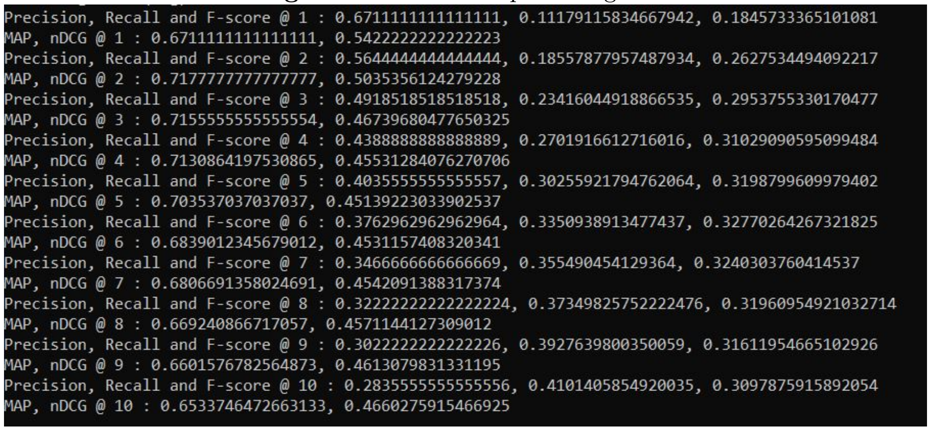


**Fig. 8.** The evaluation plot using LSA



```
Precision, Recall and F-score @ 1 : 0.6711111111111111, 0.11179115834667942, 0.1845733365101081
MAP, nDCG @ 1 : 0.6711111111111111, 0.5422222222222223
Precision, Recall and F-score @ 2 : 0.5644444444444444, 0.18557877957487934, 0.2627534494092217
MAP, nDCG @ 2 : 0.7177777777777777, 0.5035356124279228
Precision, Recall and F-score @ 3 : 0.4918518518518518, 0.23416044918866535, 0.2953755330170477
MAP, nDCG @ 3 : 0.7155555555555554, 0.467396804776503 25
Precision, Recall and F-score @ 4 : 0.4388888888888889, 0.2701916612716016, 0.31029090595099484
MAP, nDCG @ 4 : 0.7130864197530865, 0.45531284076270706
Precision, Recall and F-score @ 5 : 0.4035555555555557, 0.30255921794762064, 0.3198799609979402
MAP, nDCG @ 5 : 0.703537037037037, 0.45139223033902537
Precision, Recall and F-score @ 6 : 0.3762962962962964, 0.3350938913477437, 0.32770264267321825
MAP, nDCG @ 6 : 0.6839012345679012, 0.4531157408320341
Precision, Recall and F-score @ 7 : 0.3466666666666669, 0.355490454129364, 0.3240303760414537
MAP, nDCG @ 7 : 0.6806691358024691, 0.4542091388317374
Precision, Recall and F-score @ 8 : 0.3222222222222224, 0.37349825752222476, 0.31960954921032714
MAP, nDCG @ 8 : 0.669240866717057, 0.4571144127309012
Precision, Recall and F-score @ 9 : 0.3022222222222226, 0.3927639800350059, 0.31611954665102926
MAP, nDCG @ 9 : 0.6601576782564873, 0.4613079831331195
Precision, Recall and F-score @ 10 : 0.2835555555555556, 0.4101405854920035, 0.3097875915892054
MAP, nDCG @ 10 : 0.6533746472663133, 0.4660275915466925
```

**Fig. 9.** The evaluation metrics using LSA

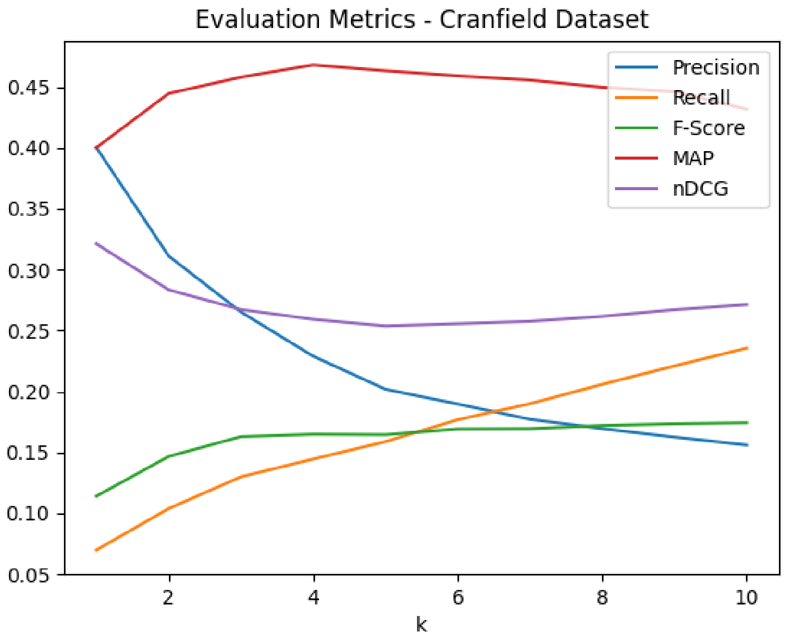
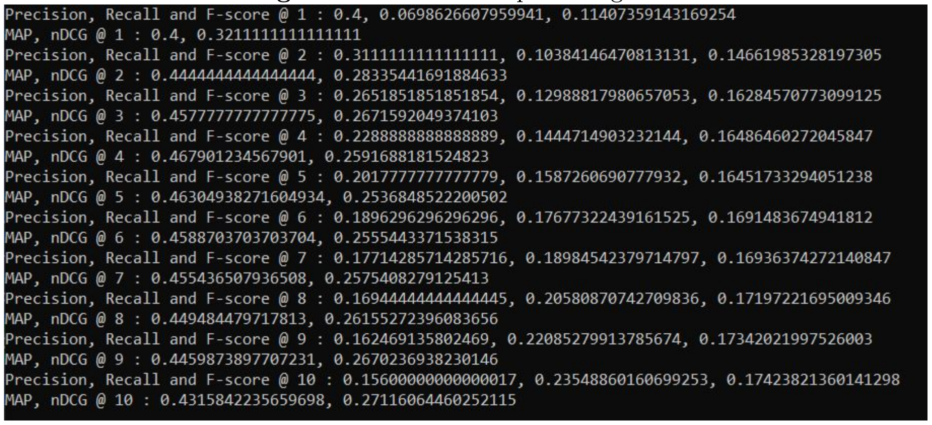


**Fig. 10.** The evaluation plot using ESA



```
Precision, Recall and F-score @ 1 : 0.4, 0.0698626607959941, 0.11407359143169254
MAP, nDCG @ 1 : 0.4, 0.3211111111111111
Precision, Recall and F-score @ 2 : 0.3111111111111111, 0.10384146470813131, 0.14661985328197305
MAP, nDCG @ 2 : 0.4444444444444444, 0.28335441691884633
Precision, Recall and F-score @ 3 : 0.2651851851851854, 0.12988817980657053, 0.16284570773099125
MAP, nDCG @ 3 : 0.4577777777777775, 0.2671592049374103
Precision, Recall and F-score @ 4 : 0.2288888888888889, 0.1444714903232144, 0.16486460272045847
MAP, nDCG @ 4 : 0.467901234567901, 0.2591688181524823
Precision, Recall and F-score @ 5 : 0.2017777777777779, 0.1587260690777932, 0.16451733294051238
MAP, nDCG @ 5 : 0.46304938271604934, 0.2536848522200502
Precision, Recall and F-score @ 6 : 0.1896296296296296, 0.17677322439161525, 0.1691483674941812
MAP, nDCG @ 6 : 0.4588703703703704, 0.2555443371538315
Precision, Recall and F-score @ 7 : 0.17714285714285716, 0.18984542379714797, 0.16936374272140847
MAP, nDCG @ 7 : 0.455436507936508, 0.2575408279125413
Precision, Recall and F-score @ 8 : 0.16944444444444445, 0.20580870742709836, 0.17197221695009346
MAP, nDCG @ 8 : 0.449484479717813, 0.26155272396083656
Precision, Recall and F-score @ 9 : 0.162469135802469, 0.22085279913785674, 0.17342021997526003
MAP, nDCG @ 9 : 0.4459873897707231, 0.2670236938230146
Precision, Recall and F-score @ 10 : 0.15600000000000017, 0.23548860160699253, 0.17423821360141298
MAP, nDCG @ 10 : 0.4315842235659698, 0.27116064460252115
```

**Fig. 11.** The evaluation metrics using ESA

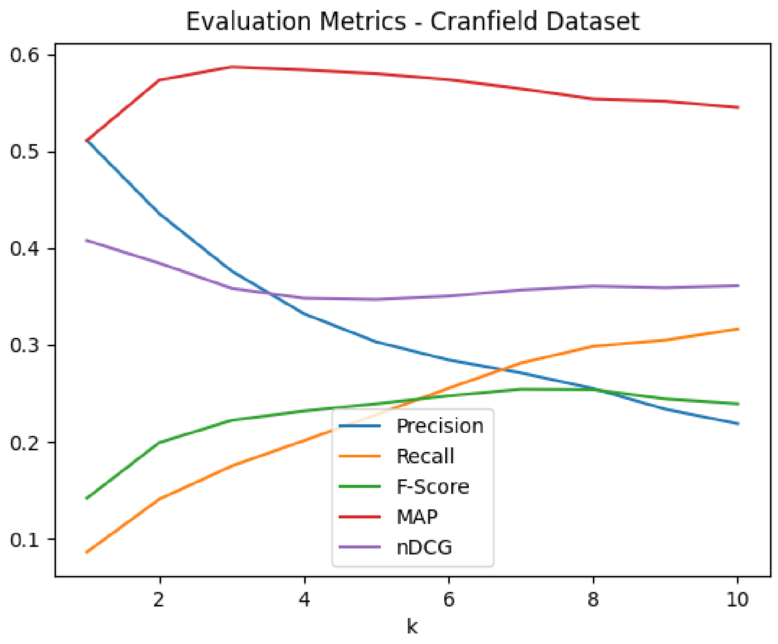


**Fig. 12.** The evaluation plot with WordNet Query Expansion

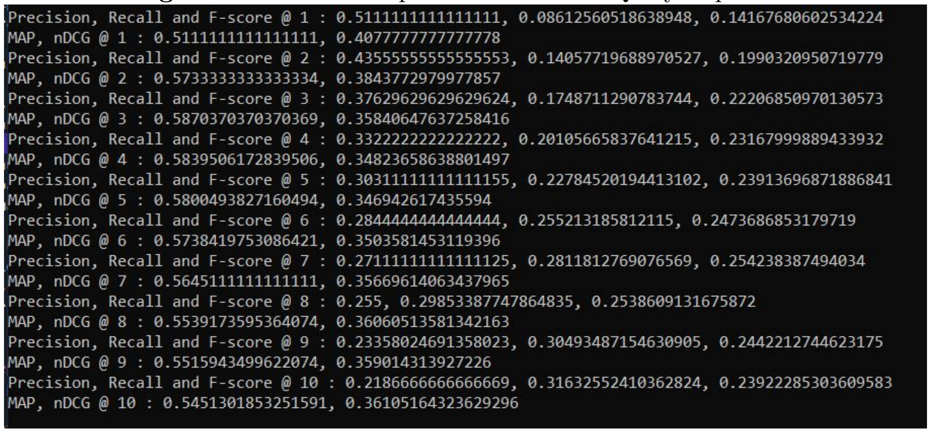

```
Precision, Recall and F-score @ 1 : 0.5111111111111111, 0.08612560518638948, 0.14167680602534224
MAP, nDCG @ 1 : 0.5111111111111111, 0.4077777777777778
Precision, Recall and F-score @ 2 : 0.43555555555555553, 0.14057719688970527, 0.1990320950719779
MAP, nDCG @ 2 : 0.5733333333333334, 0.3843772979977857
Precision, Recall and F-score @ 3 : 0.37629629629629624, 0.1748711290783744, 0.22206850970130573
MAP, nDCG @ 3 : 0.5870370370370369, 0.35840647637258416
Precision, Recall and F-score @ 4 : 0.3322222222222222, 0.20105665837641215, 0.23167999889433932
MAP, nDCG @ 4 : 0.5839506172839506, 0.34823658638801497
Precision, Recall and F-score @ 5 : 0.30311111111111155, 0.22784520194413102, 0.23913696871886841
MAP, nDCG @ 5 : 0.5800493827160494, 0.346942617435594
Precision, Recall and F-score @ 6 : 0.2844444444444444, 0.255213185812115, 0.2473686853179719
MAP, nDCG @ 6 : 0.5738419753086421, 0.3503581453119396
Precision, Recall and F-score @ 7 : 0.27111111111111125, 0.2811812769076569, 0.25423838749403
MAP, nDCG @ 7 : 0.5645111111111111, 0.35669614063437965
Precision, Recall and F-score @ 8 : 0.255, 0.29853387747864835, 0.2538609131675872
MAP, nDCG @ 8 : 0.5539173595364074, 0.36060513581342163
Precision, Recall and F-score @ 9 : 0.23358024691358023, 0.30493487154630905, 0.2442212744623175
MAP, nDCG @ 9 : 0.5515943499622074, 0.359014313927226
Precision, Recall and F-score @ 10 : 0.2186666666666669, 0.31632552410362824, 0.23922285303609583
MAP, nDCG @ 10 : 0.5451301853251591, 0.36105164323629296
```

**Fig. 13.** The evaluation metrics with WordNet Query Expansion

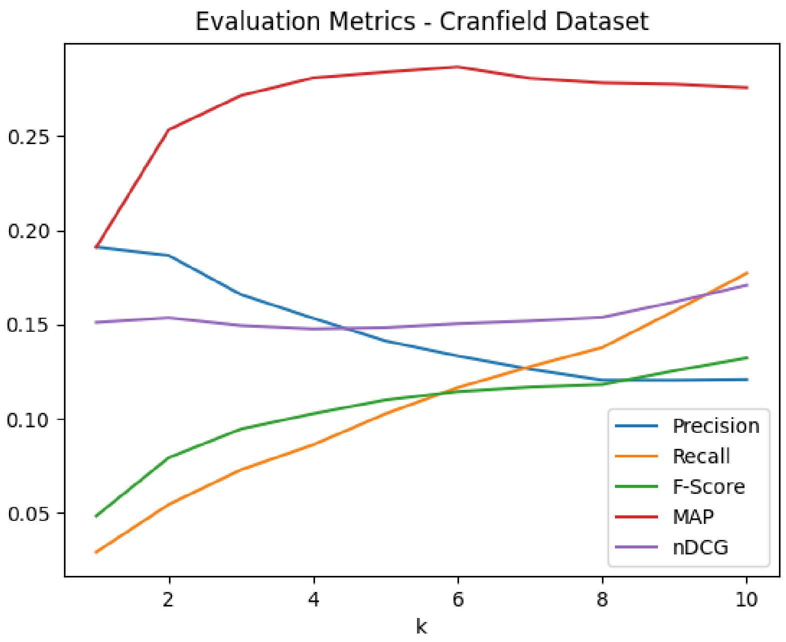
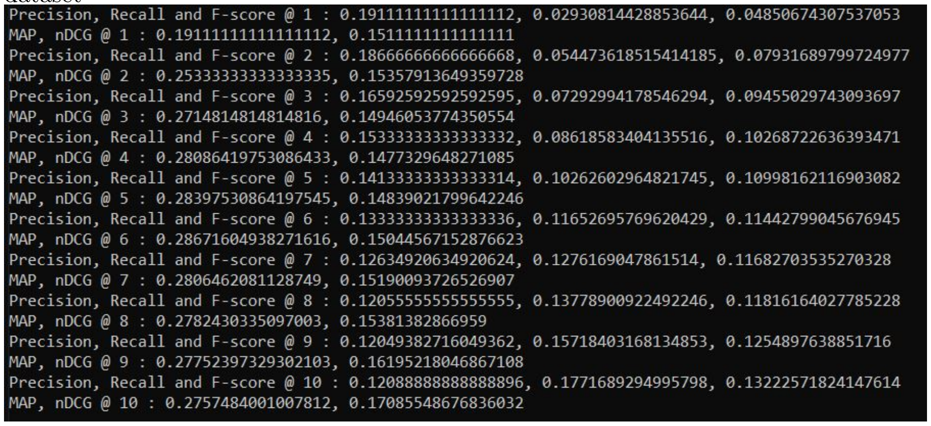


**Fig. 14.** The evaluation plot with Word2Vec embeddings trained on the Cranfield dataset



```
Precision, Recall and F-score @ 1 : 0.19111111111111112, 0.029308144288536444, 0.04850674307537053
MAP, nDCG @ 1 : 0.19111111111111112, 0.1511111111111111
Precision, Recall and F-score @ 2 : 0.18666666666666668, 0.054473618515414185, 0.07931689799724977
MAP, nDCG @ 2 : 0.25333333333333335, 0.15357913649359728
Precision, Recall and F-score @ 3 : 0.16592592592592595, 0.07292994178546294, 0.09455029743093697
MAP, nDCG @ 3 : 0.2714814814814816, 0.14946053774350554
Precision, Recall and F-score @ 4 : 0.15333333333333332, 0.08618583404135516, 0.10268722636393471
MAP, nDCG @ 4 : 0.28086419753086433, 0.1477329648271085
Precision, Recall and F-score @ 5 : 0.14133333333333314, 0.10262602964821745, 0.10998162116903082
MAP, nDCG @ 5 : 0.28397530864197545, 0.14839021799642246
Precision, Recall and F-score @ 6 : 0.13333333333333336, 0.11652695769620429, 0.11442799045676945
MAP, nDCG @ 6 : 0.28671604938271616, 0.15044567152876623
Precision, Recall and F-score @ 7 : 0.12634920634920624, 0.1276169047861514, 0.11682703535270328
MAP, nDCG @ 7 : 0.2806462081128749, 0.15190093726526907
Precision, Recall and F-score @ 8 : 0.12055555555555555, 0.13778900922492246, 0.11816164027785228
MAP, nDCG @ 8 : 0.2782430335097003, 0.15381382866959
Precision, Recall and F-score @ 9 : 0.12049382716049362, 0.15718403168134853, 0.1254897638851716
MAP, nDCG @ 9 : 0.27752397329302103, 0.16195218046867108
Precision, Recall and F-score @ 10 : 0.12088888888888896, 0.1771689294995798, 0.13222571824147614
MAP, nDCG @ 10 : 0.2757484001007812, 0.17085548676836032
```

**Fig. 15.** The evaluation metrics with Word2Vec embeddings trained on the Cranfield dataset

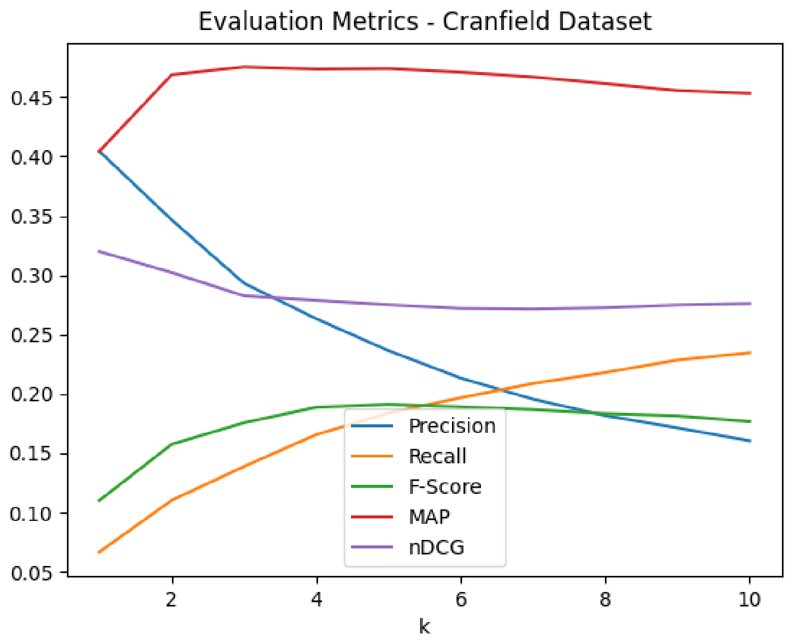
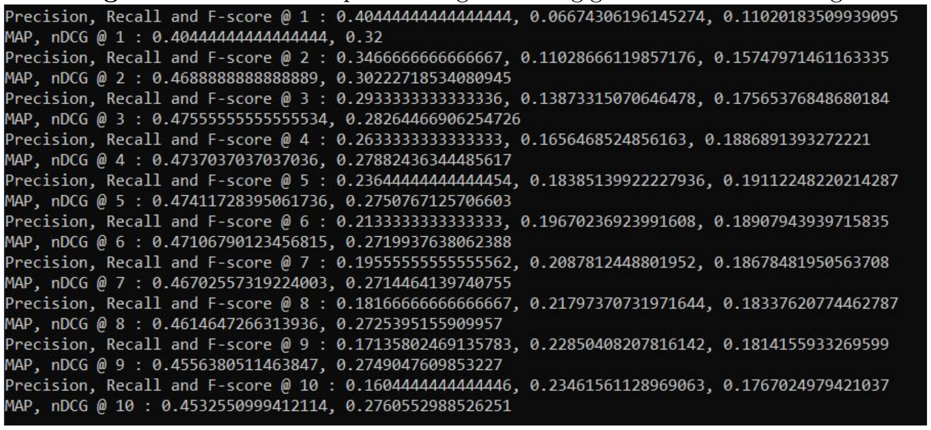


**Fig. 16.** The evaluation plot with 'glove-wiki-gigaword-200' embeddings



```
Precision, Recall and F-score @ 1 : 0.40444444444444444, 0.06674306196145274, 0.11020183509939095
MAP, nDCG @ 1 : 0.40444444444444444, 0.32
Precision, Recall and F-score @ 2 : 0.3466666666666667, 0.11028666119857176, 0.15747971461163335
MAP, nDCG @ 2 : 0.4688888888888889, 0.30222718534080945
Precision, Recall and F-score @ 3 : 0.29333333333333336, 0.13873315070646478, 0.17565376848680184
MAP, nDCG @ 3 : 0.47555555555555534, 0.28264466906254726
Precision, Recall and F-score @ 4 : 0.26333333333333333, 0.1656468524856163, 0.1886891393272221
MAP, nDCG @ 4 : 0.4737037037037036, 0.27882436344485617
Precision, Recall and F-score @ 5 : 0.23644444444444454, 0.18385139922227936, 0.19112248220214287
MAP, nDCG @ 5 : 0.47411728395061736, 0.2750767125706603
Precision, Recall and F-score @ 6 : 0.21333333333333333, 0.19670236923991608, 0.18907943939715835
MAP, nDCG @ 6 : 0.47106790123456815, 0.2719937638062388
Precision, Recall and F-score @ 7 : 0.19555555555555562, 0.2087812448801952, 0.18678481950563708
MAP, nDCG @ 7 : 0.46702557319224003, 0.2714464139740755
Precision, Recall and F-score @ 8 : 0.18166666666666667, 0.21797370731971644, 0.18337620774462787
MAP, nDCG @ 8 : 0.4614647266313936, 0.2725395155909957
Precision, Recall and F-score @ 9 : 0.17135802469135783, 0.22850408207816142, 0.1814155933269599
MAP, nDCG @ 9 : 0.4556380511463847, 0.2749047609853227
Precision, Recall and F-score @ 10 : 0.16044444444444446, 0.23461561128969063, 0.1767024979421037
MAP, nDCG @ 10 : 0.4532550999412114, 0.2760552988526251
```

**Fig. 17.** The evaluation metrics with 'glove-wiki-gigaword-200' embeddings

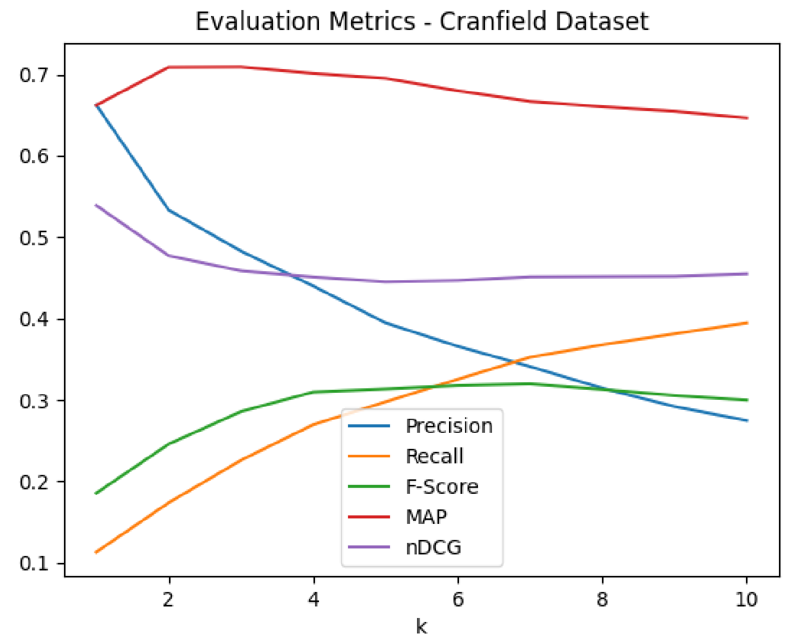
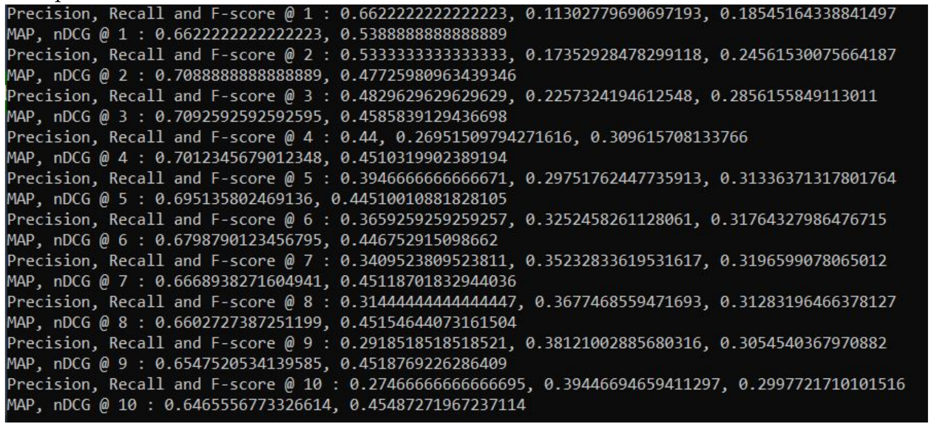


**Fig. 18.** The evaluation plot after correcting any spelling errors in the documents and the queries



```
Precision, Recall and F-score @ 1 : 0.6622222222222223, 0.11302779690697193, 0.18545164338841497
MAP, nDCG @ 1 : 0.6622222222222223, 0.5388888888888889
Precision, Recall and F-score @ 2 : 0.5333333333333333, 0.17352928478299118, 0.24561530075664187
MAP, nDCG @ 2 : 0.7088888888888889, 0.47725980963439346
Precision, Recall and F-score @ 3 : 0.4829629629629629, 0.2257324194612548, 0.2856155849113011
MAP, nDCG @ 3 : 0.7092592592592595, 0.4585839129436698
Precision, Recall and F-score @ 4 : 0.44, 0.26951509794271616, 0.309615708133766
MAP, nDCG @ 4 : 0.7012345679012348, 0.4510319902389194
Precision, Recall and F-score @ 5 : 0.3946666666666671, 0.29751762447735913, 0.31336371317801764
MAP, nDCG @ 5 : 0.695135802469136, 0.44510010881828105
Precision, Recall and F-score @ 6 : 0.3659259259259257, 0.3252458261128061, 0.31764327986476715
MAP, nDCG @ 6 : 0.6798790123456795, 0.446752915098662
Precision, Recall and F-score @ 7 : 0.3409523809523811, 0.35232833619531617, 0.3196599078065012
MAP, nDCG @ 7 : 0.6668938271604941, 0.45118701832944036
Precision, Recall and F-score @ 8 : 0.31444444444444447, 0.3677468559471693, 0.31283196466378127
MAP, nDCG @ 8 : 0.6602727387251199, 0.45154644073161504
Precision, Recall and F-score @ 9 : 0.2918518518518521, 0.38121002885680316, 0.3054540367970882
MAP, nDCG @ 9 : 0.6547520534139585, 0.4518769226286409
Precision, Recall and F-score @ 10 : 0.27466666666666695, 0.39446694659411297, 0.2997721710101516
MAP, nDCG @ 10 : 0.6465556773326614, 0.45487271967237114
```

**Fig. 19.** The evaluation metrics after correcting any spelling errors in the documents and the queries

## 8 Conclusion

### 8.1 Summary of Metrics

| Method | Precision @ 5 | Recall @ 5 | F-Score @ 5 | MAP @ 5 | nDCG @ 5 |
|---|---|---|---|---|---|
| Vector Space Model | 0.398 | 0.302 | 0.317 | 0.695 | 0.450 |
| LSA | 0.404 | 0.302 | 0.320 | 0.704 | 0.451 |
| ESA | 0.202 | 0.159 | 0.164 | 0.0.463 | 0.254 |
| Query Expansion with WordNet | 0.303 | 0.228 | 0.239 | 0.580 | 0.347 |
| Word2Vec | 0.141 | 0.103 | 0.110 | 0.0.287 | 0.148 |
| Spellcheck | 0.236 | 0.184 | 0.191 | 0.474 | 0.0.275 |

### 8.2 Summary of Hypothesis Tests

Both MAP and nDCG was tried as the metric.

| Method | p value | Confidence that new method does better baseline |
|---|---|---|
| LSA | 0.403 | 59.7% |
| ESA | 0.99 | 1% |
| Query Expansion with WordNet | 0.99 | 1% |

### 8.3 LSA

We see that among all the methods proposed, only LSA seems to reach values close to or above the vector space model. LSA with 700 topics was able to achieve scores similar or slightly higher than the vector space model. This is indicative of the fact that LSA was able to capture some of the relatedness between words, however it was not very significant. This could be due to the fact that there may be only a few words which are similar to each other or that LSA was not able to fully capture all the relations between words. For instance, LSA would not be able to capture polysemy if it exists in the document.

### 8.4 ESA

ESA doesn't do as well as expected. The reasons for this could be as follows:

- We consider only one wikipedia article per unique word in the vocabulary. It is possible that this doesn't create enough concepts to achieve any meaningful results.
- The wikipedia articles are considered orthogonal to each other. This is not a good approximation as a lot of the articles are related to each other. Hence some modification in creating relatedness among the articles could help.

### 8.5 Query Expansion with WordNet

We use Method 2 in Query Expansion - getting synoyms of each word using it's multiple synsets coupled with Method 1 for information retrieval, that is the cosine distance. We notice that Query expansion seems to drag the performance of the Vector Space model down. This is potentially due to the lack of synonyms of query words in the documents and hence the irrelevance of the expansion of the query.

### 8.6 Word2Vec

The results are relatively worse for the word2vec model trained on the cranfield dataset compared to the vector space model. This is possibly due to the fact that the size of the cranfield dataset is small. This is not enough for the neural network weights to converge to the optimal values as they usually require a much larger dataset.
The pre-trained glove vector model 'glove-wiki-gigaword-200' performs better than the word2vec model as it was trained on a much larger corpus (from wikipedia). However, since the training was not specific to the cranfield dataset, the results were still worse than the original vector space model.

### 8.7 Vector Space Model with Spellcheck

Since there weren't many spelling errors in the document or the existing queries, the results were very close. There wasn't any significant improvement, thus spelling correctiong was only used for custom queries.

## 9 References

1. **Scott Deerwester** Indexing by Latent Semantic Analysis
2. **Evgeniy Gabrilovich and Shaul Markovitch** Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis