NLP: Information Retrieval system

Riya Mahesh, Sanket Singh, and Ishaan Agarwal

Indian Institute of Technology, Madras, Chennai 600036, TN, India
{ee21b112,ee21b118,ee20b046}@smail.iitm.ac.in

Abstract. *The aim of this project is to build a robust information retrieval system using various models and compare them against different evaluation metrics on the Cranfield dataset. We start by building a simple Vector Space Model as our baseline algorithm and then employ methods based on Latent Semantic Analysis, BM25, Word2Vec and Doc2Vec intending to improvise the shortcomings of the VSM model. We further perform hypothesis testing using statistical tests to strengthen the claims about their performance.*

Keywords: Vector Space Model · LSA · Word2Vec · Doc2Vec · BM25 · Hypothesis testing

1 Introduction

Information retrieval (IR) involves finding relevant information from a large collection of data sources to satisfy a user's information need. The core challenge in IR is to accurately rank documents based on their relevance to the user's query. This project focuses on building an effective IR system for the Cranfield dataset, a commonly used benchmark in IR research.

Traditional approaches like the vector space model represent queries and documents as weighted term vectors, computing relevance scores based on vector similarities. However, these methods have limitations in handling language complexities like polysemy, synonymy, and data sparsity. To address these issues, we explore advanced techniques like Latent Semantic Analysis (LSA), Word2Vec, Doc2Vec and BM25.

By incorporating semantic analysis methods, the IR system aims to capture deeper meaning and relationships between words, improving its ability to match relevant documents beyond simple keyword matching. The BM25 ranking function provides a more robust way to score document relevance. Additionally, integrating a spell checker helps handle misspelled or malformed user queries.

This project comprehensively analyses and evaluates the proposed IR system enhancements using the Cranfield dataset. We compare the performance against a baseline vector space model across multiple ranking metrics and statistical significance tests. The goal is to develop an accurate and robust IR system to

retrieve relevant information from large document collections effectively.

2 Problem Statement

The Cranfield dataset contains 1,400 documents and 225 queries, with associated relevant documents and relevance scores for each query. The task is to build an end-to-end information retrieval (IR) system that takes a query as input and returns the top k most relevant documents from the dataset.

The developed IR system will be evaluated using standard metrics like precision, recall, f-scores, mean average precision (MAP) and nDCG. The goal is to accurately rank and retrieve the most relevant documents for a given query from the Cranfield dataset.

3 Background

The baseline model we have used is the Vector Space Model (VSM), one of the most classic and widely used models for Information Retrieval (IR). It represents both documents and queries as high-dimensional vectors in a shared vector space. The core idea behind VSM is to use the term frequencies in the documents as a basis for representing them as vectors.

In VSM, each unique term or word in the corpus is considered a separate dimension in the vector space. A document vector is constructed by computing some weighted term frequency value for each term present in that document across the respective dimensions. Common weighing schemes include:

Term Frequency (TF): The raw count of a term in a document.

Term Frequency-Inverse Document Frequency (TF-IDF): TF weighted by the inverse of the document frequency of the term across the corpus.

$$idf(word) = \log \frac{N}{n}$$

$$tf - idf(word) = tf(word) * idf(word)$$

where, N = Total number of documents in the corpus, n = total number of documents that contain the word.

We generally use a logarithmic-based IDF to diminish the impact of extremely common terms, and it tends to have a smoothing effect on the term weight distribution. The TF-IDF weighting assigns higher weights to terms that are frequent

in a document but rare across the corpus, providing a better discrimination signal.

Once the document vectors are constructed, queries are also mapped to the same high-dimensional vector space using a similar TF-IDF weighing of query terms.

To rank documents for a given query, the VSM computes the cosine similarity between the query vector and each document vector. Documents with vectors closer to the query vector (higher cosine similarity) are considered more relevant and ranked higher.

The baseline performance is as shown in Fig.1

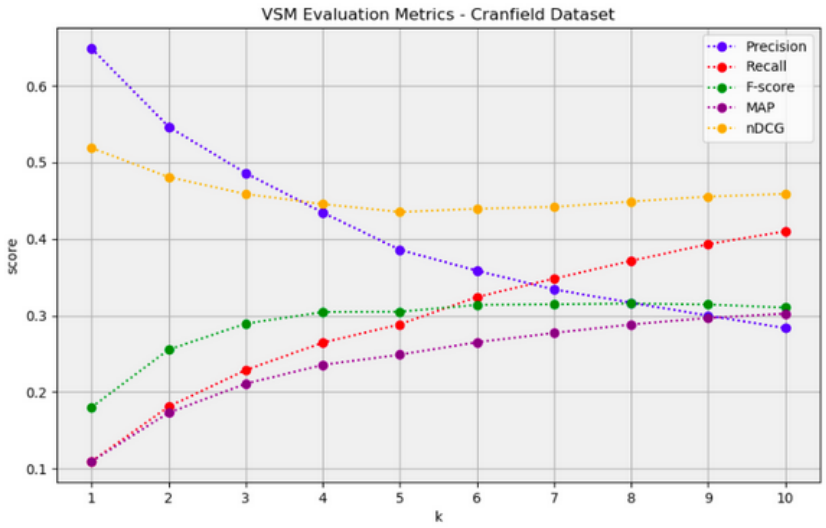

Fig. 1: Metrics under baseline (VSM)

4 Approach

Before attempting to improve over the baseline, we need to analyze its limitations:

1. Relevant documents that do not have many matching terms are not retrieved.
2. VSM assumes that the words are independent and orthogonal to each other.
3. It does not capture the semantic relationship or the context between words to retrieve documents.

4. Vocabulary mismatch problems due to polysemy and synonymy of words.
5. It ranks the documents based on query terms without considering their proximity within the document.
6. High latency & computational cost:- The TF-IDF vector is long (size of the vocabulary) with most terms having small values as they are present frequently or are absent. The process for computing similarity over such long vectors is expensive.

As an attempt to overcome these limitations, we start with minor adjustments in the pre-processing of the data. The following models have been proposed:

1. Latent Semantic Analysis (LSA)
2. Word2Vec
3. Doc2Vec
4. Best Match 25 (BM25)

We believe that the dataset is domain-specific, centred in the domain of *aerospace*. Hence, methods like ESA would not be a suitable approach as ESA relies on a general Wikipedia corpus rather than a domain specific corpus.

5 Improvements in Preprocessing

The baseline algorithm uses the following preprocessing steps:

1. **Sentence segmentation:** Punkt Sentence Tokenizer (NLTK)
2. **Tokenization:** Penn Treebank tokenizer (NLTK)
3. **Inflection Reduction:** Porter's stemmer
4. **Stopword removal:** Top-down stopwords list from NLTK

Starting off, we notice that document lengths ('*body*') are not very large (~ 200). Hence, we include the document's title to its body to enhance the context for search operations.

Following that, we use an alternative to stemming for inflection reduction. We perform lemmatization using spaCy. Lemmatization reduces words into their meaningful base form and can potentially match terms better.

Lemmatization is quite costly compared to stemming in terms of runtime, as illustrated in Table 1

Inflection reduction method	**Query preprocessing time**
Stemming	~ 0.1 seconds
Lemmatization	~ 90 seconds

Table 1: Query preprocessing runtimes

6 Latent Semantic Analysis

VSM picks out documents with the *same* words as the query. It disregards the meanings of the words themselves. Hence, it fails to retrieve relevant documents that represent the same concept as the query but do not share many common words. It also assumes a space of independent orthogonal words, which is rarely the case so. When we attempt to overcome this, we arrive at the circularity problem-

Two words are similar if they appear in the same document, two documents are similar if they contain similar words.

LSA solves the circularity problem by performing Singular Value Decomposition (SVD) on the term-document matrix(1).

$$A = U\Sigma V^T$$

where U = Left Singular Vector, Σ = Singular values, V = Right Singular Vector

We perform a smoothing such that we retain the k-largest singular values which represent the dimensions of the latent space. In essence, we are representing our term document matrix in a smaller space. This results in a rank-k approximation of the original matrix

$$A_k = U_k \Sigma_k V_k^T$$

A_k is the best k-rank approximation to A in the least squares sense. An advantage of this approach is it gives us both a term and a document vector. The new k-dimensional query vector D_q can be computed using-

$$D_q = A_q^T U_k \Sigma_k^{-1}$$

where A_q is the term vector for the query.

The performance using the LSA model is as shown in Fig.2

7 Word2Vec

LSA handles the retrieval of conceptually similar documents. The method, however, does not consider the context in which the words appear. In an attempt to improve this, we selected word2vec as our next method.

Word2vec is a widely used unsupervised technique that generates word embeddings for every unique word appearing in the corpus. Word embeddings are vector representations of words that capture their semantics and contextual usage. The primary idea is to build a neural network that learns to predict a word, given its context words in a chosen window frame or vice-versa.

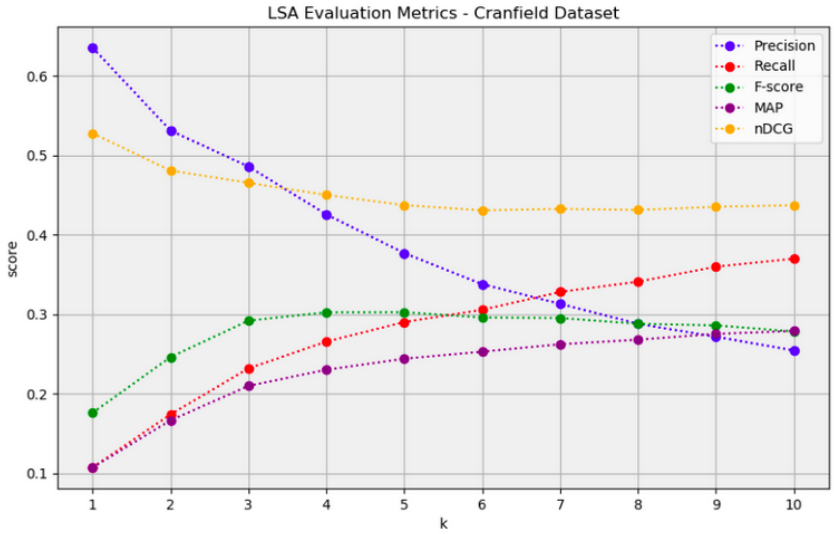

Fig. 2: Metrics under LSA *(with improved preprocessing)*

The underlying principle is based upon the *Distributional Hypothesis*, which states that words with similar meanings appear in similar contexts. The method can be implemented using one of the below architectural frames(2):

1. **Continuous Bag of Words:**
   - In this method, the model takes in one-hot encoded vectors of context words as inputs. These vectors are projected onto the next neural network layer, where each context word is reduced to its embedding.
   - The next layer accumulates these either through averaging/concatenation. This averaged embedding represents the word embedding for the target word in the continuous vector space.
   - It is then passed through the output layer with softmax activation function to generate a final vector for the target word as a probability distribution over the terms in the vocabulary. This vector is trained against the actual one-hot encoded representation of the target word.

2. **Continuous Skip Gram model:**
   - The skip-gram model uses the one-hot encoded vector of the target word as input and predicts the context words that appear around it.
   - The input vector is projected onto the next layer, where it is reduced to its embedding representation.
   - It is passed to the output layer with softmax distribution to get the final vectors for all the context words as a probability distribution over the terms in the vocabulary.

– The model is trained with pairs of target words and their context words in the window frame.

Some of the advantages of word2vec are listed below:

1. Word2vec accounts for the context of words and hence captures semantic similarities and associations between words in the window.
2. The neural network can be trained efficiently over a large corpus of data.

The limitations are enumerated below:

1. While word2vec considers the local context in which the word appears, it fails to capture broader global semantic associations.
2. In the presence of words that have not appeared in the corpus before, the word embedding generated may not be meaningful. This is handled properly in the case of baseline, as new words that appear in the query are not considered while representing the query as a vector of vocabulary terms.

The performance of the Word2Vec model is as shown in Fig.3

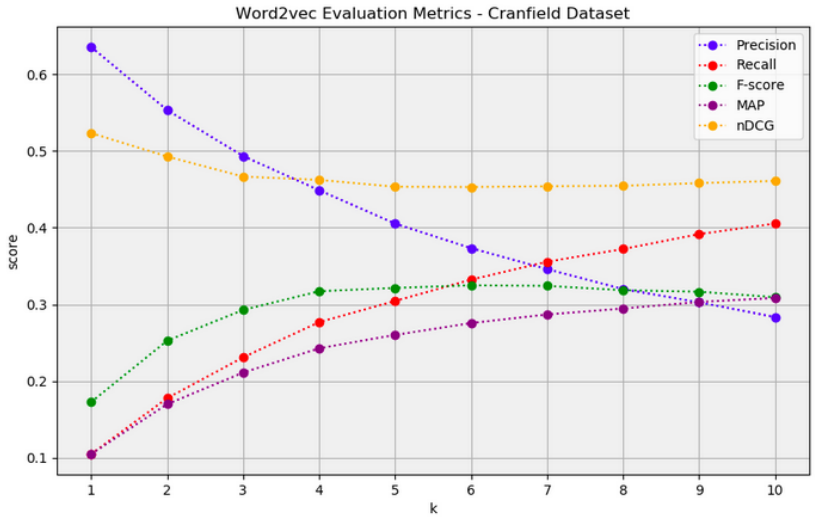

Fig. 3: Metrics under Word2Vec *(with improved preprocessing)*

8 Doc2Vec

Doc2vec is an unsupervised method that generates document embeddings, which train to predict the words that belong to the document. The doc2vec algorithm

is an extension of word2vec approach and is more useful in capturing global semantics and concepts represented by the document as a whole, rather than just word embeddings that are generated using context words within a chosen window frame.

Unlike word2vec, doc2vec uses one hot encoded vectors for both words and document IDs. Word vectors are encoded using the vocabulary terms as a basis, and every document is assigned a unique document tag with the document IDs as a basis. The doc2vec model has two different architectures:

- **Distributed Memory:** In this architecture, the neural network takes the one-hot encoded document ID vector (tag) and the vector representations of the context words as input. These are projected onto the next layer, generating embeddings for document and context words followed by their concatenation. It is then passed through the output layer with softmax activation function to generate a final vector for the target word as a probability distribution over the terms in the vocabulary(3).
- **Distributed Bag of Words:** In this method, the document ID is taken as input to generate a document embedding. The embedding is passed through several neural network layers along with softmax activation at the output layer. The output layer results in vector representations for randomly sampled words from the document(4).

The primary advantage of doc2vec is its ability to comprehend semantic relations at the document level. It can also be used to train large corpus but the training can be computationally expensive. Like word2vec, unseen words are also assigned embeddings that may not be accurate.

The performance of the Doc2vec model is as shown in Fig.4

9 Best Match 25

A clear limitation of the VSM model was the disregard for document length, inherently being biased towards larger documents. Longer documents tend to contain occurrences of more terms, which can introduce bias. Hence, it fails to retrieve relevant documents of smaller length because of fewer matching words with a query. *Document length normalization* offsets this bias by dividing the term frequency by the document's length and applying a normalization factor.

BM25 (Best Match 25) picks up from VSM - instead of using the traditional *TF*IDF*, it incorporates the length of the document to compute the BM25 score.

For a query Q, containing terms $q_1, \ldots, q_n$, the BM25 score of a document D is:

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}\left(q_i\right) \cdot \frac{f\left(q_i, D\right) \cdot \left(k_1 + 1\right)}{f\left(q_i, D\right) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)},$$

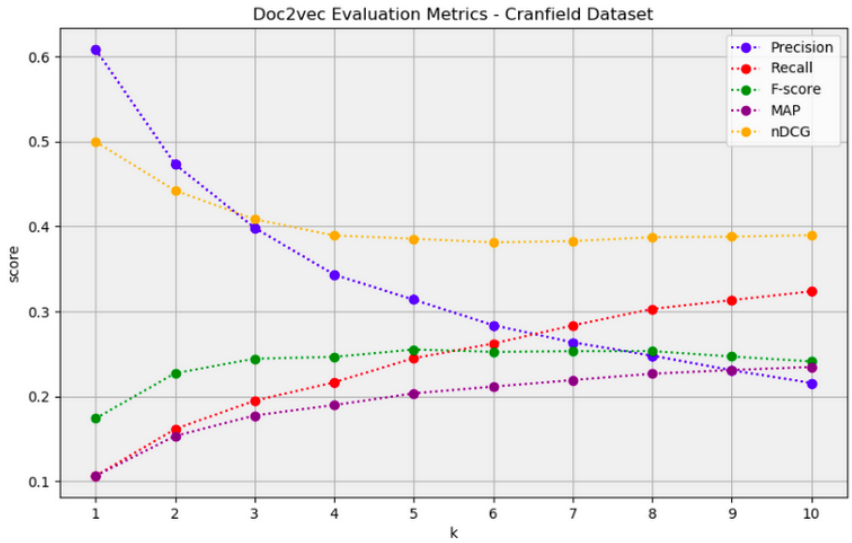

Fig. 4: Metrics under Doc2Vec *(stemming)*

where:

- $f(q_i, D)$ = term frequency of q_i in D.
- $|D|$ = length of the document D
- avgdl = average length of document in text collection
- k_1 and b are free parameters

Some advantages of BM25 are as follows:

1. It considers both the frequency of terms and normalization of document length, which helps in unbiasing against larger documents.
2. Due to its simplicity, BM25 is scalable for large datasets.

The limitations of BM25 are listed below:

1. Similar to the baseline, BM25 is a bag of words model. It treats documents as unordered words, ignoring the context and semantic relationship between the words.
2. It only retrieves documents having the same words as the query.

The performance of the BM25 model is as shown in Fig5

10 Experimentation

The Cranfield dataset contains 1400 documents and 7406 unique tokens across the corpus. The dataset has two blank documents (doc-id = 471, 995) which

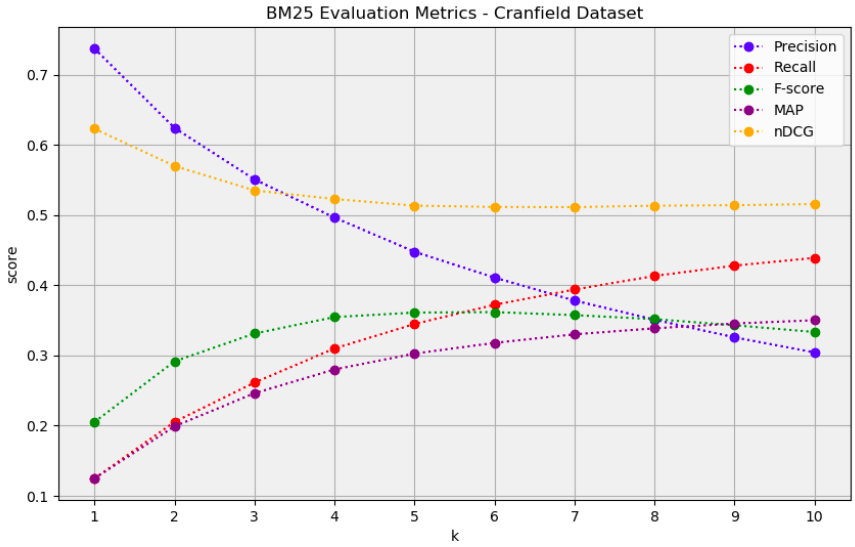

Fig. 5: Metrics under BM25 *(with improved preprocessing)*

are retrieved at the end of all the methods. We choose a simple Vector Space Model built using inverted indexing as our baseline algorithm. The unique terms occurring across all documents are chosen to be the basis of the vector space. All documents and queries are represented as vectors in this space using tf-idf as explained before. The documents are ranked and retrieved based on cosine similarity. We further compare the retrieval results against the ground truth and evaluate precision, recall, F-score, mean average precision and nDCG for all the methods.

10.1 LSA

Our LSA model is built for a term-document matrix of size 1400×7406. We perform the SVD of this matrix and retain the top 400 singular values when arranged in descending order to obtain a reduced space representation. The rank of the reduced dimension space 'k' was chosen as 400 through experimentation as we obtained the highest scores across most of the metrics.

10.2 Word2Vec

We are training the pre-built Gensim Word2Vec model with our dataset and then using this to generate embeddings for words in the query. The following are the phases in the model-

Building Vocabulary: The model takes input data and constructs a vocabulary by assigning an index position to every unique term in the corpus. This is

used to generate one-hot encoded vectors for every word.

Training: Gensim provides options to train using *Continuous Bag of Words (CBOW)* or *Skip-Gram* algorithms. We considered the skip-gram algorithm as it is known to work well with smaller datasets and can give more accurate representations for infrequent words (5) . We have chosen a word embedding size of 400 and a context window size of 2 before and after the target word. The default negative sampling value of 5 has been retained, which implies that for every target word - context words pair, 5 randomly sampled words that do not appear within the window frame are considered as negative samples. This helps the model differentiate positive and negative associations of words. These parameters have been tuned to achieve the best results across different evaluation metrics.

Retrieval: Once the embeddings for all words are obtained, we now represent the document embedding as the mean of the word embeddings of all the words occurring in it. We follow a similar approach for generating query vectors and then compute cosine similarity between the document and query vectors to rank them.

10.3 Doc2Vec

For doc2vec we are using Gensim Doc2Vec to train our dataset using the *Distributed Bag of Words* approach. The model stages are similar to that of Word2Vec - building vocabulary, training and retrieval.

We have chosen a document embedding size of 400 and a context window frame of size 2. We are using a learning rate of 0.135 and training the model for 500 epochs. We use the trained model to obtain document vectors for new queries and compute cosine similarity between queries and documents to rank them.

10.4 BM25

We use the BM25 framework from (6) retaining the default parameters of k1 = 1.5, b = 0.75 as they produced the best evaluation measures.

11 Evaluating Robustness

To compare the performances of our improvised algorithms over baseline, we conduct hypothesis testing fixing one evaluation measure at a time. Let μ_d represent the true mean difference (or population mean difference) for any evaluation measure for algorithms A_1 and A_2.

We consider the following hypotheses:

$$\text{Null } (H_0) : \mu_d = 0$$

$$\text{Alternative } (H_1) : \mu_d \neq 0$$

We shall use the *paired t-test*(7) to decide between the null and the alternative hypothesis.

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\left(s^2\left(\frac{1}{n_1} + \frac{1}{n_2}\right)\right)}}$$

The test assumes that the dependent variable is normally distributed. From the t-value, we arrive at the p-value:

For the test statistic $T = t$ is in one instance. The lowest significance level at which the null will be rejected for is said to be the p-value of the sampling.

We run the above test for A_1 = VSM and A_2 = model

t-test is suitable for samples whose size is less than 30. (8) There are 225 queries in the dataset. We will split them into sets of 9 queries each. Thus, we collect 25 samples for each method. We arrive at the t and p values by retrieving top-10 documents for each query. Lower p-value indicates higher confidence in rejecting null.

12 Results

We now present the evaluation scores for each approach against standard IR measures for top-1 and top-10 retrievals. *(title)* means that we have included the title with the body while preprocessing. *(title+lemma)* means that we further performed lemmatization instead of stemming in the preprocessing phase. The results have been tabulated in Table 2. For each metric, the best-performing algorithm has been marked in bold.

Fig. 6 - 10 show the plots for different IR metrics across all the approaches against the baseline.

The t-values and p-values for each model against the baseline have been tabulated in Table 3.

Models	Precision		Recall		F1-Score		MAP		nDCG	
	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10
VSM (Baseline)	0.648	0.283	0.109	0.409	0.179	0.310	0.109	0.302	0.518	0.458
VSM (title)	0.653	0.291	0.109	0.416	0.179	0.317	0.109	0.313	0.515	0.467
VSM (title+lemma)	0.671	0.299	0.114	0.430	0.187	0.326	0.114	0.320	0.554	0.485
LSA	0.600	0.234	0.102	0.341	0.168	0.257	0.102	0.256	0.488	0.408
LSA (title)	0.626	0.250	0.106	0.368	0.175	0.276	0.106	0.275	0.504	0.427
LSA (title+lemma)	0.635	0.254	0.106	0.369	0.175	0.278	0.106	0.278	0.527	0.437
Word2Vec	0.622	0.285	0.101	0.413	0.167	0.312	0.101	0.304	0.492	0.456
Word2Vec (title)	0.613	0.282	0.100	0.410	0.165	0.309	0.100	0.302	0.49	0.452
Word2Vec (title+lemma)	0.635	0.284	0.104	0.406	0.172	0.310	0.104	0.308	0.516	0.461
Doc2Vec	0.591	0.202	0.103	0.308	0.169	0.226	0.103	0.224	0.488	0.378
Doc2Vec (title)	0.608	0.215	0.106	0.323	0.173	0.240	0.106	0.234	0.500	0.389
Doc2Vec (title+lemma)	0.551	0.212	0.096	0.316	0.156	0.236	0.096	0.225	0.458	0.381
BM25	0.706	0.289	0.120	0.420	0.197	0.317	0.120	0.332	0.586	0.493
BM25 (title)	0.706	0.292	0.119	0.428	0.196	0.321	0.119	0.339	0.592	0.499
BM25 (title+lemma)	**0.737**	**0.304**	**0.124**	**0.439**	**0.205**	**0.333**	**0.124**	**0.350**	**0.623**	**0.515**

Table 2: Evaluation scores against standard IR metrics

Models	Precision@10		Recall@10		F1-Score@10		MAP@10		nDCG@10	
	t-value	p-value	t-value	p-value	t-value	p-value	t-value	p-value	t-value	p-value
LSA	5.574	9e-06	4.877	5e-05	5.303	1e-05	4.471	1e-04	4.774	7e-05
Word2Vec	1.076	0.292	0.441	0.663	0.807	0.427	1.306	0.310	1.021	0.317
Doc2Vec	8.301	1e-08	7.598	7e-08	7.944	3e-08	8.506	1e-08	7.314	1e-07
BM25	-0.136	0.892	-1.304	0.204	-0.611	0.546	-3.679	0.001	-3.695	0.001

Table 3: Statistical test results

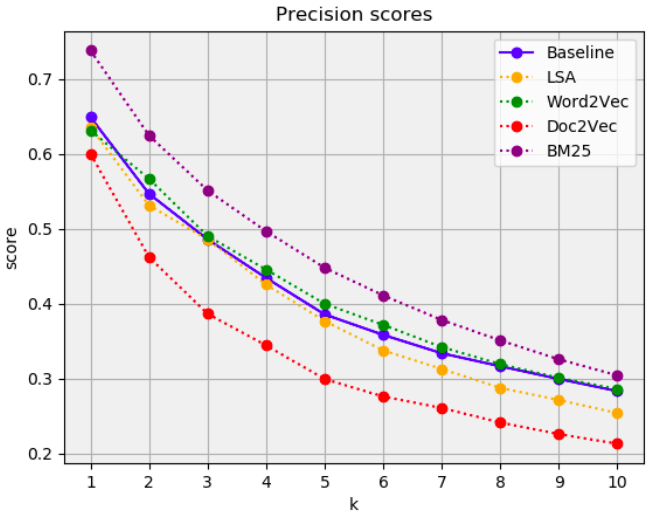

Fig. 6: Precision scores

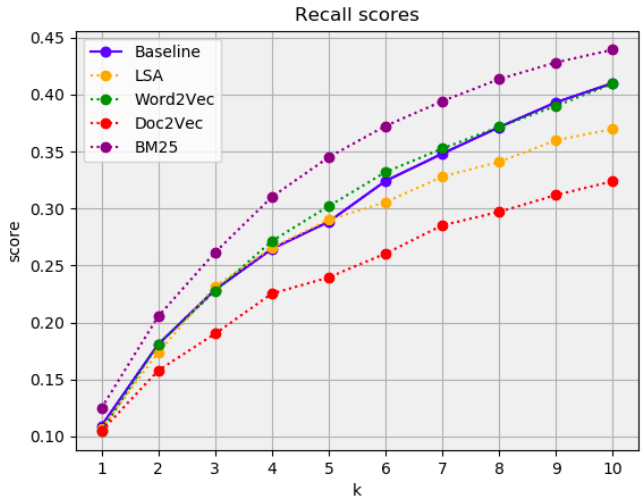

Fig. 7: Recall scores

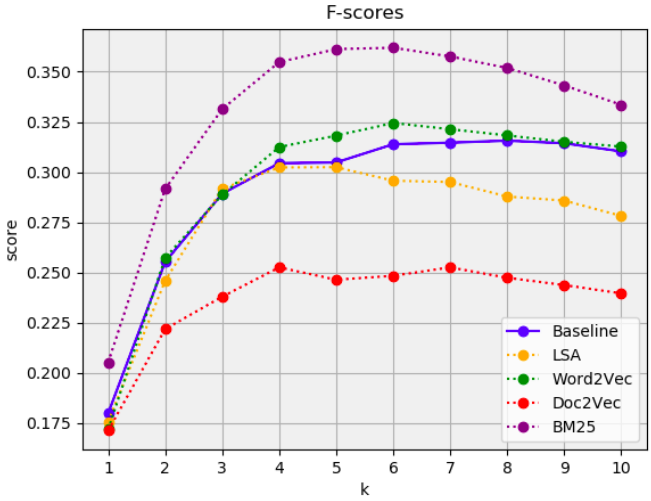

Fig. 8: F-scores

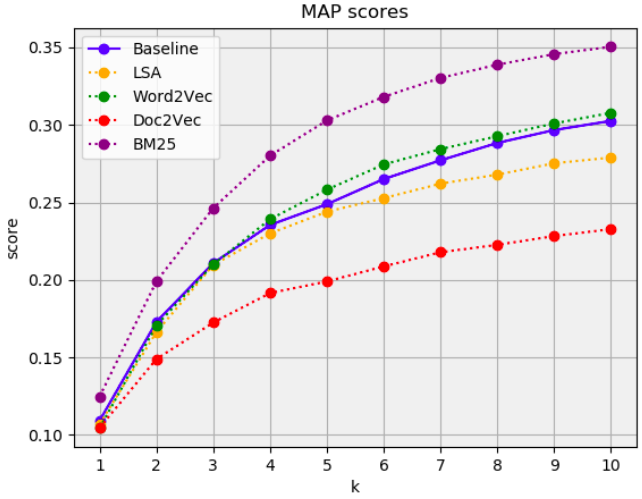

Fig. 9: MAP scores

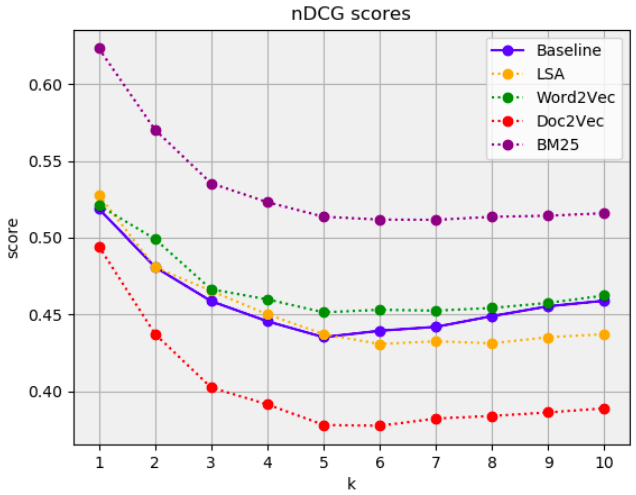

Fig. 10: nDCG scores

13 Observations

The following observations were made from the results obtained:

1. BM25 outperforms the baseline and has better scores across all metrics among all the models-
   - Among the methods, a negative t-value is obtained only for BM25 across all measures.
   - MAP and nDCG are more qualifying measures for an IR system as they take ranking into account as well. We obtain very good p-value (≤ 0.05) for BM25 against the baseline for both these measures.
2. We get better scores with the inclusion of title irrespective of the method for almost all the metrics, with very marginal increase in the runtime of the IR system leading to good scalability and is suitable for large datasets as well.
3. Using lemmatization instead of stemming gives slight improvement of the scores for most methods, but it comes at the cost of runtime. Hence, it is not very scalable and is not suitable for large datasets as the difference in performance is only marginal.
4. Although LSA did not perform better than baseline, the reduced vector representations of documents and queries are fairly accurate. We are able to obtain nearly on par results by saving computational costs and memory.
5. Documents 471 and 995 are empty (have no title or body) and are retrieved at the end for all methods except LSA. The original term doument matrix would have a column of 0s corresponding to those documents but on performing SVD and reducing the rank, the new matrix obtained is smoothened and these columns tend to have non-zero entries leading to non-zero cosine similarity values. This can be visualised as a failure case of LSA.

6. Word2Vec shows slight improvement across all measures from the baseline and the reduced embeddings proved effective in representing the documents and queries saving computational cost.
7. Doc2vec did not perform better than the baseline algorithm. We suspect this is because the cranfield dataset is not sufficiently large enough for training these deep learning models.

14 Conclusion

By thorough analysis, we show that BM25 is better than the baseline (Vector Space Model) with respect to all standard IR measures (Precision, Recall, F1-score, MAP, nDCG) in the task of information retrieval on the cranfield dataset with a confidence interval of 99.9% under MAP and nDCG. The cranfield dataset has relevance scores of documents for each query. MAP and nDCG are suitable measures as they take ranking into account as well. Word2Vec performs on par with the baseline (p-value $\sim$ 0.3). The baseline outperforms both LSA and Doc2Vec (p-value $\sim$ 1e-06).

Bibliography

[1] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990. [Online]. Available: https://psychology.uwo.ca/faculty/harshman/latentsa.pdf

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013. [Online]. Available: https://arxiv.org/abs/1301.3781

[3] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*. PMLR, 2014, pp. 1188–1196. [Online]. Available: http://proceedings.mlr.press/v32/le14.pdf

[4] K. Mani, I. Verma, H. Meisheri, and L. Dey, "Multi-document summarization using distributed bag-of-words model," in *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE, 2018, pp. 672–675. [Online]. Available: https://arxiv.org/pdf/1710.02745

[5] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013. [Online]. Available: https://arxiv.org/abs/1310.4546

[6] "Rank-bm25: A two line search engine," https://pypi.org/project/rank-bm25/.

[7] "Paired t-tests," https://www.scribbr.com/statistics/t-test/.

[8] "Samples in t-test," https://www.statisticssolutions.com/t-test/.