CS6370: Project Report

Sumanth Doddapaneni[1*] **Raghavan AK**[2*] **Aswanth Kumar M**[3*]

IIT Madras
{cs21d409[1], cs21s025[2], cs21m010[3]}@smail.iitm.ac.in

Abstract

Our previous Information Retrieval (IR) system used TF-IDF approach to index documents and cosine similarity to judge the relevance of queries with documents. We evaluated the IR system on Cranfield dataset using various precision-recall and ranking based metrics. To improve upon the existing system we hypothesised new systems (i) use context aware embeddings, (ii) address polysemy by moving towards sentence representations (iii) reduce vector sparsity (iv) address synonymy with query expansion. In this work we implement the above mentioned strategies and compare the results against our existing IR system. We also perform ablations on these experiments and draw insights from them. Our code is available on GitHub[1].

1 Introduction

Our current IR system uses TF-IDF (Qaiser and Ali, 2018) based indexing which treats the documents and vectors as a bag of words. While we were able to achieve decent performance with the model, there are were queries for which not even a single document was retrieved correctly in the top-10. Improvements can be made moving towards more sophisticated word/sentence representation techniques. Our current model has many drawbacks (i) it has very high dimensional sparse vectors which leads to low cosine scores (ii) low recall due to the usage of synonyms in queries and documents (iii) low precision due to the bag of words representation of sentence. In this project we focus on addressing these issues. We first identify the the issues in TF-IDF indexing, hypothesize the issues and state the possible solutions to address the issues. We will be using the Cranfield dataset to investigate our ideas.

In order to increase the quality of returned documents and to improve recall, we used Query expansion by selecting some synonyms in the query words and adding them to the query. We later performed Latent Semantic Analysis (LSA) (Deerwester et al., 1990) to obtain the compressed representations of the documents. LSA is a distributional semantics technique for analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms.

Next we move towards pre-trained embeddings - both word and sentence level. We used the pretrained embeddings from word2vec (Mikolov et al., 2013) for word level embeddings and Sentence-Transformer (Devlin et al., 2019) for sentence level embeddings. In order to reduce retrieval times we cache the index and observe a 80% improvement in the retrieval speeds.

Later, we implemented a spell correction mechanism using the methods taught in class i.e. generating candidates using bigrams and words that are one-edit distance apart. The spell correction module tries to correct the query that is either present in the Cranfield dataset or entered by the user. This helps the IR system to get the appropriate results. We found that spell correction has an accuracy of 81.61% on one letter spelling errors using the bigrams method.

2 Dataset

For this project we work on Cranfield dataset. The dataset contains documents related to automobile aerodynamics. The dataset contains 1400 documents which will be used to index and retrieve based on the queries. The dataset also comes with 225 queries which can be used to evaluate the performance of the system. The dataset also provides human relavance scores for the queries *w.r.t* the documents. These scores will be used for ranking the nDCG evaluation metric.

* All authors have contributed equally.
[1] https://github.com/sumanthd17/CS6370

3 Evaluation Metrics

Evaluation measures for an information retrieval system are used to assess how well the search results satisfied the user's query intent. Such metrics are often split into kinds: online metrics look at users' interactions with the search system, while offline metrics measure relevance.

3.1 Precision

In information retrieval, precision is the fraction of retrieved documents that are relevant to the query:

$$precision = \frac{|relevant \cap retrieved|}{|retrieved|} \quad (1)$$

3.2 Recall

Recall is defined as the fraction of the relevant documents that are successfully retrieved.

$$precision = \frac{|relevant \cap retrieved|}{|relevant|} \quad (2)$$

3.3 F1-Score

The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean.

$$precision = \frac{2 \times precision \times recall}{precision + recall} \quad (3)$$

3.4 MAP

Average Precision at k (AP@k) is the sum of precision@K for different values of K divided by the total number of relevant items in the top K results.

The Mean Average Precision at k (MAP@k) measures the average precision@K averaged over all queries.

$$MAP@k = \frac{\sum_{q=1}^{Q} AveP(q)}{|Q|} \quad (4)$$

3.5 nDCG

The Normalized Discounted Cumulative Gain for k shown recommendations (nDCG@k) divides this score by the maximum possible value of DCG@k for the current user, *i.e* what the score DCG@k would be if the items in the ranking were sorted by the true relevance. This is called Ideal Discounted Cumulative Gain (IDCG@k)

$$nDCG_p = \frac{DCG_p}{IDCG_p} \quad (5)$$

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_p} - 1}{log_2(i+1))} \quad (6)$$

$$IDCG_p = \sum_{i=1}^{|REL_p|} \frac{rel_i}{log_2(i+1))} \quad (7)$$

4 Methods

4.1 Bag of Words

Bag of words is a text modeling technique and can also be said that it is a method of feature extraction with text data. It is a simple and flexible way of extracting features from documents.

This representation of text describes the occurrences of words within a document. Also, just keeps the track of word counts and disregard the word order and semantic relationships. Any information about the order or structure of words in the document is completely discarded. By using the bag of words model, variable-length texts can be converted to fixed-length vectors.

We use the bag of words model to represent both queries and documents as vectors where queries and documents are preprocessed. Data preprocessing involves removal of tokenization, stemming/lemmatization and stop word removal. We computed the TF-IDF (Qaiser and Ali, 2018) (term-frequency and Inverted document frequency) matrix along with add-one smoothing. The same is applied to a query. Then we computed the cosine similarity for a query and the TF-IDF. The documents are arranged in the decreasing order of their cosine similarity where the higher the cosine similarity the more similar the document and query are.

Bag of words doens't consider the semantic relationship. In order to overcome this issue we used word embeddings. Also, the TF-IDF matrix is sparse which involves lot of computation. To overcome this issue we tried LSA to get important features from the Cranfield dataset.

4.1.1 Query Expansion

Query expansion is a process of expanding a given query to improve retrieval performance. It involves expanding the search query to match additional documents using techniques like finding synonyms of words and searching for the synonyms as well.

Wordnet has the concept of synsets, where a synset is a grouping of synonymous words that express the same concept. Some of them have only

one synset and some have several. We tokenized each query and got synonyms for each token using wordnet synsets and appended the query with the synonyms obtained. This expanded the query and we tried to get better search results.

The synsets in wordnet are arranged in a way that most used synsets are indexed with a lower index. We tried query expansion with sysnet[0] (most frequently used synset) and synsets till index 3. The values we obtained at k=10 when experimenting with them is at Table 3.

The Cranfield dataset is very specific to the domain of Aeronautics. Based on the above results we found that the query expansion using wordnet work well a bit when compared with to the naive vector space model. We later explored LSA methods.

Method	P	R	F1	MAP	nDCG
synsets - 0	0.30	0.42	0.32	0.66	0.49
synsets - 0, 1, 2	0.28	0.41	0.31	0.63	0.46
without query expansion	0.29	0.42	0.32	0.67	0.48

Table 1: Evaluation scores of various methods on the Cranfield dataset using query expansion

4.1.2 LSA

Using LSA (Deerwester et al., 1990), we obtained compressed representations of documents by generating a low-rank approximation of the term-document matrix. Based on the assumption that the words that are similar occur in similar pieces of text, using LSA we captured the inherent relationships between the meanings of terms. We used a mathematical technique called Singular Value Decomposition (SVD) to get the term-document matrix T is computed by:

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

Where columns of U represent the concepts, where each concept is represented in terms of words. U is a column orthogonal matrix. Σ consists of singular values arranged in descending order. Each Row of V^T is a concept represented in terms of documents. V^T is a row orthogonal matrix.

LSA reduces the dimensionality of vectors by using a smaller set of k concepts to represent documents. Where the 1st k components (k largest eigenvalues in Σ) from each row of V^T represent each document. In order to get the updated query mapped to the new subspace formed by the k concepts, we used:

$$q_k = \Sigma_k^{-1} U_k^{-1} q$$

Where q_k is the new query representation and q is the actual TF-IDF of the query.

Then we computed the cosine similarity between the updated query and documents using k concepts and ordered the documents based on their similarity.

In the below section of "Rank of LSA" we showed how to obtain the optimal value of k. We found that the optimal value of k is around 200. Based on further experimenting, at k = 220 we obtained the best results for LSA.

4.1.3 LSA with weighted Wordnet

We wanted to use Wordnet to get word wise relatedness and weight the documents accordingly.During the indexing time we take each word from each doc and compute the similarty between them with all the other words in corpus. Using this similarity value, we add weighted sum to document representation. This enriches the document representation with sematic related words present in corpus.

We use WuPalmer – WordNet Similarity as

$$wup(s1, s2) = 2\frac{depth(LCS(s1, s2))}{depth(s1) + depth(s2)}$$

In Wordnet every word is represented as Synsets, so to compute the similarity between two words, we compute the similarity of corresponding Synsets of words.

Once the documents representation is enriched with similarity, are use LSA technique explained in previous section to get the top k components and use this condensed representation to index the documents.In our experiments we found k value of 200 to giving the best performance.

We also do the same enrichment to query documents before querying it with index.

4.2 Embeddings

We have seen that the bag-of-words type of representation doesn't capture the semantic relationship between the words. So we move towards vector representations that try to capture these relationships.

4.2.1 Word Embeddings

We use the Word2Vec (Mikolov et al., 2013) embeddings for representing the queries and documents of the corpus. Word2Vec provides us 300-dimensional vectors for each word and we use the

aggregated information of all the words as the representation of the query and document. We average all the word embeddings in a query / document and consider that as the sentence presentation. We try further variations which is discussed in Section 7

Word2Vec is trained on the Google News dataset Corpus . The Cranfield data contains very technical terms which are not present in the training corpus used for training Word2Vec. As a result we skip the words which are not present in Word2Vec vocabulary when creating the aggregated sentence representation.

Table 2 shows some examples of the words which are dropped when creating the sentence representations.

In-order to overcome this issue we tried to train a Word2Vec model on the Cranfield dataset. But Word2Vec algorithm is quite data hungry and the results obtained with Word2Vec trained on Cranfield dataset are much lower compared to LSA method.

4.2.2 Sentence Embeddings

Word2Vec models generates embeddings that are context-independent: *i.e* - there is just one vector representation for each word, different senses of the word are combined into one single vector. While, BERT (Devlin et al., 2019) generates context-dependent embeddings, *i.e* it allow us to have multiple vector representations for the same word, based on the context in which the word is used.

We use the embeddings from SentenceBERT (Reimers and Gurevych, 2019) which is a Siamese style BERT trained to provide sentence representations. We use the embedding of the [CLS] token as the sentence representation for the entire sentence.

5 Spelling Correction in IR

In our IR system, we implemented two methods for spelling correction. In both methods, if a word is present in the corpus we ignore it as we assume it is a valid word. Spell correction is performed only on the invalid words.

In the first method, we used the bigrams approach where we generated bigrams for all the words in the vocabulary and created a bigram reverse index where each bigram is mapped to the list of words that have the bigram in it. Later, for the invalid word we generate the bigrams, and using the bigram reverse index we get all words that share the same bigram. These form the candidate words. To reduce the candidate words, we consider candidate words whose bigrams have 50% overlapping with the invalid word. Then, we compute the edit distance with these candidate words and consider the replacement of the invalid word which has the least edit distance. We used the Levenshtein distance as the edit distance.

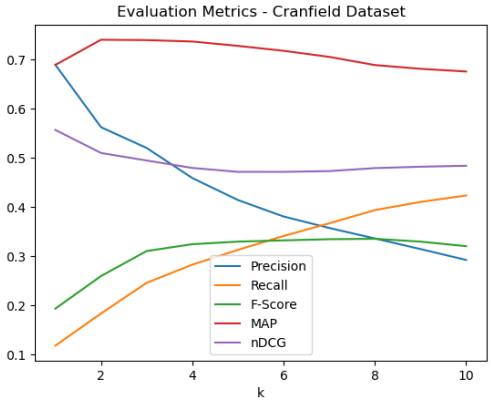

Figure 1: The plot shows the scores of Baseline model on all the evaluation metrics (at each rank)

In the second method, we generate a list of all candidate words that are at an edit distance of 1 from the invalid word. For these candidate words, we compute the probability of the word as below:

$$p(w) = \frac{count(w_q)}{count(w_c)}$$

where w_q is the number of times the word is present in the query and w_c is the number of times the word is present in the corpus. The word with the highest probability is considered as the replacement for the incorrect word. Since the candidate words are present in the corpus, add-one smoothing can be avoided.

6 Results and Discussion

Table 3 shows the summary of the results of all the methods discussed above.

The Cranfield dataset is a highly technical dataset on aerodynamics domain. Any generic word representation scheme struggles to catch the essence of the documents and queries. This is due to large number of homonym words present in corpus. Due to this, any word embedding model will retrieve in right ranking documents for queries. Compared to word embedding based representation of documents, sentence embedding represention perform better. This is due to better collective meaning being produced for a document.

Query	Missing Words
in practice, how close to reality are the assumptions that the flow in a hypersonic shock tube using nitrogen is non-viscous and in thermodynamic equilibrium .	non-viscous, lift-drag
can non-linear shallow shell analysis be reduced to an engineering technique by use of the matrix .	non-linear
how do large changes in new mass ratio quantitatively affect wing-flutter boundaries .	wing-flutter
are asymptotic methods sufficiently accurate in the determination of pre-buckling stresses in torispherical shells, or must we resort to numerical methods .	pre-buckling, torispherical

Table 2: Words missing in the Word2Vec vocabulary

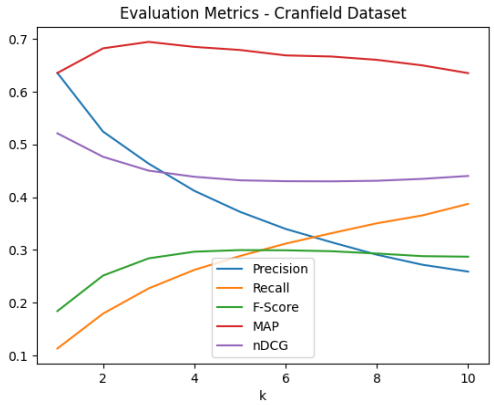

Figure 2: The plot shows the scores of Sentence Embedding model on all the evaluation metrics (at each rank)

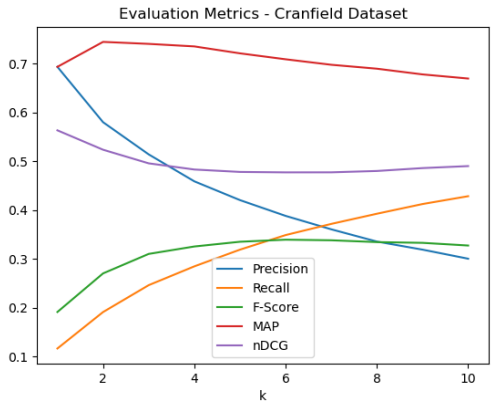

Figure 3: The plot shows the scores of Query Expansion on all the evaluation metrics (at each rank)

Method	P	R	F1	MAP	nDCG
Baseline	0.29	0.42	0.32	0.67	0.48
Query Exp.	0.30	0.42	0.32	0.67	0.49
LSA	0.25	0.37	0.27	0.58	0.41
Wordnet	0.28	0.41	0.32	0.62	0.46
Word Emb.	0.10	0.13	0.10	0.29	0.12
Sent. Emb.	0.25	0.38	0.28	0.63	0.44

Table 3: Evaluation scores of various methods on the Cranfield dataset

Due to this highly technical domain, methods such as query expansion, weighted wordnet with LSA performs better.

Both the query expansion and weighted wordnet enriches the document representation in very similar way. While the Query expansion method brings similar meaning words from wordnet in the documents and queries, the weighted wordnet method enriches the documents and queries by adding weighted word represention of related words (using wordnet synsets).

7 Ablations

7.1 Rank of LSA

There are three methods that can be utilized to find the value of k for the LSA. The methods are scree plot, plot metric evaluation, and plot of the cumulative proportion of variance. we experimented with the above methods to get the best value for k.

In the first method, the scree plot is plotted by using the singular values obtained after performing SVD vs the component number. The component number at which there is a sudden change in the gradient can be taken as k which is also reffered to as component number where the knee of the plot occurs. The same plot is shown in Figure: 4. From the plot we can see that the knee occurs in around 200.

In the second method, we plotted the evaluation metric measures like Precision@10, Recall@10

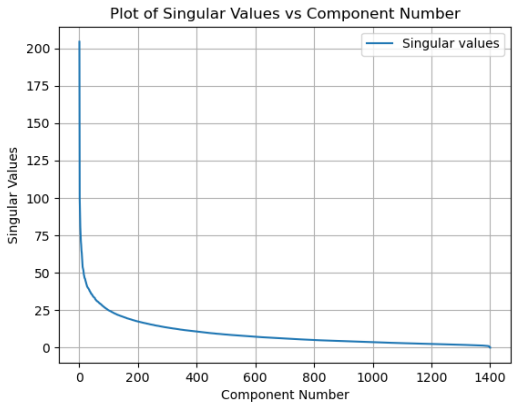

Figure 4: LSA: The plot shows singular values against component number

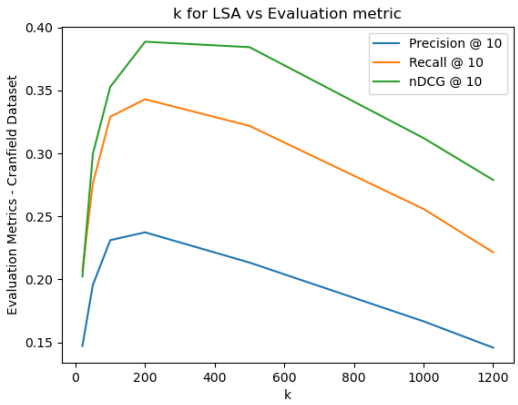

Figure 5: LSA: The plot shows evaluation metric values against component number

and nDCG@10 against the k components like [20, 50, 100, 200, 500, 1000, 1200]. Other measures like MAP, F-score, etc., can also be considered. Using the values obtained above and against the component number we plotted a plot and observed that at k = 200 the metric measures have the best values compared to other k values. The same plot is shown in Figure: 5.

In the third method, we plotted the cumulative proportion of variance against the component numbers. The same plot is shown in Figure: 6. The proportion of variance is calculated as below where s_k is the k^{th} singular value:

$$pov(k) = \frac{s_k^2}{\sum_{i=1}^{I} s_i^2}$$

The c_k (Cumulative variance @ k) is calculated as below:

$$c_k = \sum_{i=1}^{k} pov(i)$$

Generally, in this plot, we need to consider at least 70% of the variance. We found that at around k=200, this condition is satisfied.

From the above experiments, we concluded that around rank k=200, yeilded the best results for the LSA.

7.2 Pre-trained vs Trained Word2Vec Embeddings

Since many of the important terms are missing in both queries and documents are missing from the Word2Vec vocabulary, we decided to train a Word2Vec model on the Cranfield dataset. But due to the very small size of the cranfield data corpus, the learned representations are quite bad.

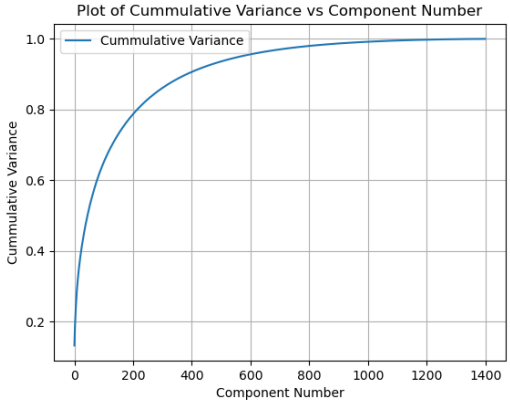

Figure 6: LSA: The plot shows cumulative variance against component number

The results of the experiments are shown in Table 4

Method	P	R	F1	MAP	nDCG
Pretrained	0.10	0.13	0.10	0.29	0.12
On Cranfield	0.02	0.02	0.02	0.07	0.02

Table 4: Pre-trained Word2Vec vs Word2Vec trained on Cranfield

Method	P	R	F1	MAP	nDCG
Baseline	0.29	0.42	0.32	0.67	0.48
Basel + Title	0.30	0.43	0.33	0.67	0.49

Table 5: Adding title information while indexing the documents

7.3 Adding Title Information

One of the ablation study we performed is by adding the title information when we are indexing the document. This seems to help the vector space models (5) as we are giving more weight to the word vectors. But similar performance gains are not observed in the contextual embeddings. This is because the context based networks are normalizing the effect of weighting the sentence.

7.4 When does the spelling correction module go wrong?

We analyzed the performance of the spelling correction module with a customized evaluation dataset and found the module works with an accuracy of 81.61% using the bigrams method. These were some failures we came across with the Cranfield dataset queries even though the query words were correct. The same is shown in Table: 6.

- The words are not present in the corpus but other valid words which have very less edit distance from these words are being replaced as the system assumes that the input word is misspelled and the spelling correction module tries to get to replace it with the best word. This isn't a true failure as the module isn't aware of the input word as it is not present in the corpus.

While using our evaluation dataset for the spelling correction module, we observed more failure cases. The same is presented in the Table: 7.

1. The incorrect word "porposed" in the table is corrected to "composed" as the Cranfield dataset consists of "composed" and it has the maximum overlapping bigrams with "porposed". The same applies to words incorrect words "ivnerse" and "variable".

2. The incorrect word "fluctuatign" in the table is corrected to "fluctuation" and not "fluctuating" as both of them are equally likely. The same is the case with the incorrect word "kirchhfofs". The part of speech is not captured because the spell correction module doesn't consider the context of the word.

8 Conclusion

After trying different methods, we found that query expansion along with more weightage to title gave the best results to IR system. The contextual embeddings methods are failing as the dataset is very domain specific. Even though we performed spelling correction, it didn't improve the IR performance as the our spelling correction model doesn't consider context of words and many terms in queries are not present in the Cranfield dataset documents.

Correct Word (not in corpus)	Corrected To
invert	inert
trust	trust
kink	sink
stop	step
establishes	established

Table 6: Spelling correction of correct words on the Cranfield dataset

Incorrect Word	Spell correction by module	Actual word
porposed	composed	proposed
ivnerse	diverse	inverse
vairable	available	variable
fluctuatign	fluctuation	fluctuating
kirchhfofs	kirchhoff	kirchhoffs

Table 7: Spelling correction of correct words on the our Evaluation dataset

References

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *J. Am. Soc. Inf. Sci.*, 41(6):391–407.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Shahzad Qaiser and Ramsha Ali. 2018. Text mining: Use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications*, 181(1):25–29.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and*

the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pages 3980–3990. Association for Computational Linguistics.