Information Retrieval System

Aadarsh, Amizhthni, Fardeen, Leshya and Sudarshana

Indian Institute of Technology, Madras, Chennai 600036, TN, India
{EE21B001,EE21B015,EE21B046,EE21B078,EE21B142}@smail.iitm.ac.in

Abstract. This project focuses on developing a robust information retrieval system by implementing and comparing multiple retrieval models on the Cranfield dataset. We begin with a basic Vector Space Model (VSM) to serve as a performance baseline. Building upon that, we explore more advanced approaches, including Latent Semantic Analysis (LSA), CRN, BM25, Word2Vec, Word2Vec with LDA and Doc2Vec — each selected to address specific limitations of the VSM. To rigorously evaluate these models, we assess their effectiveness using a range of retrieval metrics and apply statistical hypothesis testing to validate the significance of performance differences. Abstract Here

Keywords: Information Retrieval System · Vector Space Model · Latent Semantic analysis · Case Retrieval Net · Word2Vec · Word2Vec with Latent Dirichlet Allocation · BERT · BM25 · Doc2Vec · Hypothesis Testing

1 Introduction

"Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)." - from the book Introduction to information retrieval by *Manning et al.*. The task of an IR system is to represent the documents effectively and rank them based on their relevance to the user's query.

The classic model for an IR system, the vector space model is considered as the baseline model. Some other models - Word2Vec, Doc2Vec, Word2Vec+LDA, LSA, CRN, Bert and BM25 have been implemented and their performance with the Cranfield Dataset is compared.

By integrating semantic analysis techniques, the information retrieval system is designed to understand deeper relationships between words, enabling it to identify relevant documents beyond basic keyword matching. The use of the BM25 ranking function further enhances the system's ability to score document relevance more effectively. This project conducts a thorough evaluation of these enhancements using the Cranfield dataset, comparing their performance against a baseline Vector Space Model across various ranking metrics. Statistical significance tests are also employed to ensure the reliability of the results. The overarching aim is to build a precise and reliable retrieval system capable of efficiently extracting relevant information from large-scale document collections.

2 Problem Statement

We are required to build an end-to-end information retrieval (IR) system that performs effective and efficient document ranking on the Cranfield dataset—a curated collection of 1,400 scientific abstracts and 225 queries. Each query is associated with a set of relevant documents and corresponding relevance scores. The system accepts a query as input and returns the top $k$most relevant documents. The IR system will be rigorously evaluated using standard retrieval metrics, including precision, recall, F-scores, Mean Average Precision (MAP), and nDCG, with the ultimate goal of accurately ranking and retrieving the most relevant documents from the corpus. We use various methods including hybrid methods that combine knowledge from linguistic, background, and introspective sources to represent documents. Some examples are Latent Semantic Analysis (LSA) and Explicit Semantic Analysis (ESA). We need to test the hypotheses rigorously using appropriate hypothesis testing methods.

3 Background

In the classic Vector space model, documents and queries are represented as vectors of terms in the document in a multidimensional space, called the Term Document matrix. The similarity between a document and the input query is computed using cosine similarity between their vector representations. The frequency of occurrence of a term within a document is denoted by its term frequency (tf) and the differentiating property of a term is denoted by the inverse document frequency (idf). With the weight based representation of the documents and queries, the cosine similarity is determined and ranked to find the relevant documents.

The baseline model (which uses the vector space formalism) assumes independence and orthogonality between all words. This is not a valid assumption. On top of that, the vectors are very large and sparse because of the size of the vocabulary, as a result of which, it takes a considerable amount of time to compute the cosine similarity.

The other shortcomings of this model include:

- Failure to capture polysemy and synonymy causes vocabulary mismatch problems.
- The term document matrix is very large and computing similarity is computationally very intensive. Moreover, addition of a new document requires us to recalculate all the vectors.
- The sequential ordering in which the terms appear in a document is lost in the vector space representation.
- It has limited conceptual understanding and doesn't capture semantic relationship between words. It just represents documents as a linear combination of words and is purely a lexical matching approach.

As seen in this graph of the evaluation metrics of the VSM, the Baseline method performs rather poor.

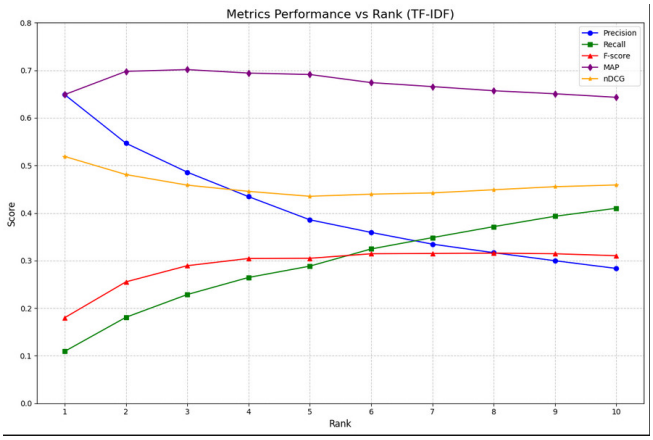

Fig. 1: VSM evaluation metrics

4 Approach

- We have built upon the baseline model, which includes a few pre-processing steps namely, Sentence Segmentation, Word Tokenisation, Stopword removal, lemmatization; and evaluate the model performance based on the most frequently used metrics in the field
- We have implemented the LSA approach, CRN-based PMI approach; along with the feature of auto-completion of queries, based on a very similar tf-idf approach, Word2Vec, Doc2Vec, Word2Vec+LDA, LSA, CRN, BERT and BM25.
- We are not including methods like ESA as they rely on a general Wikipedia corpus rather than a domain specific corpus and would not be suitable for the Cranfield data

4.1 Word2Vec

- While Latent Semantic Analysis (LSA) captures high-level conceptual similarities between documents, it does not consider the actual **context in which words appear**. To address this limitation, we explore **Word2Vec**, a powerful unsupervised technique that generates context-aware vector representations **(embeddings)** for every unique word in a corpus.
- At the core of Word2Vec lies the Distributional Hypothesis, which posits that words appearing in similar contexts tend to have similar meanings. Word2Vec operationalizes this idea by training a shallow neural network to

either predict a word given its context (CBOW) or predict context words given a target word (Skip-gram).

We have tried out both these methods and chosen the best one. **Architectures:**

- **Continuous Bag of Words (CBOW):**
  - Takes one-hot encoded vectors of surrounding context words as input.
  - These are projected into a lower-dimensional embedding space.
  - The projected embeddings are averaged (or concatenated) to form a representation of the context.
  - A softmax output layer then predicts the target word, training the network to produce embeddings that best reconstruct the missing word.
- **Skip-Gram:**
  - Takes the one-hot encoded vector of a target word as input.
  - Projects it into its embedding vector in the hidden layer.
  - Uses a softmax output layer to predict each of the surrounding context words.
  - Trains the model using (target, context) word pairs extracted from a window around each word in the corpus.

In both approaches, the hidden layer representation is considered the final word embedding.

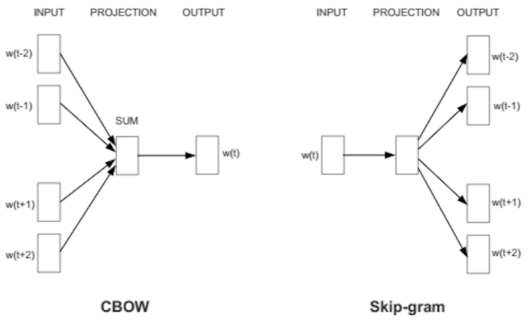

Fig. 2: W2V methods

Practical Considerations:

- The embedding dimension (number of hidden units)
- The window size around each word
- The number of training epochs
- The choice between CBOW and Skip-gram

Present Implementation: In our IR system, we use Word2Vec to generate document and query embeddings by averaging the word embeddings across each document or query. These embeddings are then compared using cosine similarity to rank documents based on relevance.

4.2 Doc2Vec

- Doc2Vec offers a powerful extension to Word2Vec by generating vector representations not just for individual words, but for entire documents.
- This makes it particularly well-suited for tasks like information retrieval, where the goal is to assess the relevance of entire documents to a given query rather than just identifying word-level similarities.
- Unlike Word2Vec, which only captures local context through a fixed-size window of neighboring words, Doc2Vec incorporates document-level semantics—allowing it to model broader concepts, themes, and the overall meaning conveyed by a document.
- **Methodology:**
  - It does this by associating each document with a unique identifier (tag) and training the model to predict words using both document IDs and context words.
  - In the **Distributed Memory (DM) architecture**, Doc2Vec learns to predict target words based on the document tag and surrounding words
  - In the **Distributed Bag of Words (DBOW)** variant, it learns to predict randomly sampled words using only the document vector.

  Both the architectures were tried and DBOW has been used in the report further, due to its better performance.

4.3 Word2Vec along with LDA

- Word2Vec does not consider the overall structure or topic of the document. Two documents may have completely different word usage patterns but still have the same topic (e.g., 'global warming' and 'climate crisis').
- LDA (Latent Dirichlet Allocation), in contrast, captures global semantics. It represents each document as a mixture of latent topics, where each topic is a distribution over words. This gives a high-level interpretable structure to documents.

This hybrid model provides:

- Fine-grained word-level context
- Coarse-grained document-level topic structure
- This is especially useful in retrieval systems where queries might be vague, multi-faceted, or semantically complex.
- **Distributional Hypothesis powers Word2Vec:** "You shall know a word by the company it keeps."
- **Generative topic modeling powers LDA:** Documents are mixtures of topics; topics are mixtures of words.

The hybrid model merges:

- Neural representation learning (continuous space)
- Probabilistic topic modeling (discrete latent space)
- This enhances robustness, recall, and semantic matching, especially for:
  - Short queries
  - Ambiguous or polysemous words
  - Sparse data situations

4.4 LSA

Latent Semantic Analysis (LSA) is a technique in natural language processing that overcomes the limitations of traditional Vector Space Models (VSM). While VSM retrieves documents by matching exact terms, it disregards the meanings of the words, failing to identify documents that discuss the same concepts using different vocabularies. This leads to what is known as the circularity problem, expressed as:

> *Two words are similar if they appear in the same document; two documents are similar if they contain similar words.*

LSA resolves this by mapping both terms and documents to a latent concept space, where similarity is measured using higher-order associations. Mathematically, it uses Singular Value Decomposition (SVD) on the term-document matrix A:

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

Here, U and V are the left and right singular vectors representing terms and documents respectively, and Σ is a diagonal matrix of singular values. To reduce dimensionality and noise, we retain only the k largest singular values and their corresponding vectors, resulting in the rank-k approximation:

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

This low-rank approximation preserves the most significant latent semantic structure of the corpus, allowing for better generalization. The reduced representation not only smooths out noise and sparsity in the original matrix, but also captures synonymy (similar meanings with different words) and polysemy (same word with multiple meanings) more effectively than VSM.

In this reduced latent space, both terms and documents are represented as dense vectors. A query can similarly be projected into this space using:

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

where A_q is the term vector for the query. This enables comparison of query and document vectors within the same semantic space.

LSA also brings computational advantages by compressing large sparse matrices into a smaller space, reducing the need for extensive storage and improving retrieval performance. Since LSA is built on the assumption that similar meanings will co-occur in similar textual contexts, it effectively captures the hidden relationships among words and documents, making it a powerful tool for concept-based information retrieval.

4.5 Case Retrieval Net (CRN) based Pointwise Mutual Information (PMI)

In the framework of Case Retrieval Networks (CRNs), **Information Entities (IEs)** serve as the fundamental units of knowledge. These entities may encapsulate atomic pieces of information, such as attribute–value pairs. A case within this structure is defined as a collection of such IEs, and the case base itself is structured as a network. This network includes nodes representing individual IEs and additional nodes representing the complete cases. The network employs two types of connections: similarity arcs that link related IE nodes, and relevance arcs that connect a case node to its constituent IEs.

The conceptual design of IEs and their interrelations within a CRN can be linked to the notion of **Pointwise Mutual Information (PMI)**, particularly in the way PMI captures semantic associations and co-occurrence patterns between terms in a corpus.

PMI is a statistical measure used to estimate the strength of association between two words based on their joint and individual probabilities of occurrence.

$$\text{PMI}(w_1, w_2) = \log\left(\frac{P(w_1, w_2)}{P(w_1) \cdot P(w_2)}\right) \tag{1}$$

Here, $P(w_1, w_2)$ denotes the probability of words w_1 and w_2 occurring together, while $P(w_1)$ and $P(w_2)$ represent the independent probabilities of the individual words.

A higher (positive) PMI score implies that the word pair co-occurs more frequently than would be expected by random chance, indicating a strong semantic association. Conversely, a negative PMI value suggests that the words appear together less frequently than expected under independence.

In various Natural Language Processing (NLP) applications—such as information retrieval, text classification, and sentiment analysis—PMI is often utilized to identify meaningful word relationships. By capturing such semantic links, PMI contributes to feature extraction and can enhance the performance of machine learning algorithms.

4.6 Bert

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a machine learning framework developed by Google for natural language processing (NLP) tasks. In the context of IT, BERT is a foundational technology that enables computers to better understand human language by analyzing the context of words in both directions (left and right of each word), rather than just one direction as in earlier models.

Key Features of BERT

- Bidirectional Contextual Understanding: BERT considers the full context of a word by looking at the words before and after it, which allows for a much deeper understanding of language nuances and ambiguities.

- Transformer Architecture: BERT is based on the transformer model, which uses self-attention mechanisms to weigh the importance of each word in a sentence relative to others.
- Pre-trained and Fine-tuned: BERT is first pre-trained on large text corpora (like Wikipedia), then fine-tuned for specific IT or business applications such as search, chatbots, or document classification

4.7 BM25

BM25 (Best Matching 25) is a ranking function used in information retrieval to estimate the relevance of documents with respect to a search query. It belongs to the family of probabilistic retrieval models and is derived from the Probabilistic Relevance Framework (PRF). BM25 is widely used in search engines and forms a strong baseline in many modern retrieval systems.

BM25 Scoring Formula

For a document d and a query $q = \{q_1, q_2, \ldots, q_n\}$, the BM25 score is computed as:

$$\text{BM25}(q, d) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, d) \cdot (k_1 + 1)}{f(q_i, d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{\text{avgdl}})} \tag{2}$$

Where:

- $f(q_i, d)$: frequency of term q_i in document d
- $|d|$: length of document d (in number of terms)
- avgdl: average document length in the collection
- k_1: term frequency saturation parameter (commonly set to 1.2)
- b: document length normalization parameter (commonly set to 0.75)
- $\text{IDF}(q_i)$: inverse document frequency of term q_i

Key Properties of BM25

Feature	BM25 Capability
Term Frequency	Yes; higher term frequency increases the relevance score, but with saturation
Document Length Normalization	Yes; long documents are penalized relative to the average document length
Term Importance (IDF)	Yes; rare terms are weighted more heavily than common terms
Training Requirement	None; BM25 is unsupervised and requires no training data
Use Case	Fast and interpretable baseline for ranking documents; useful in search engines and zero-shot IR

Table 1: Summary of BM25 Properties

Table 2: Comparison between BM25 and BERT-based Retrieval Models

Aspect	BM25	BERT-based Models
Type	Lexical (bag-of-words)	Neural, contextual semantic
Matching Mechanism	Exact keyword matching	Semantic matching using contextual embeddings
Input Representation	Sparse vectors (TF-IDF, term frequency)	Dense, contextualized embeddings
Speed	Very fast, suitable for real-time use	Slower (esp. cross-encoder); faster variants like bi-encoder exist
Indexing	Simple inverted index	Dense vector indexing (e.g., using Faiss or ANN structures)
OOV Handling	Poor (no embeddings for unseen words)	Handles unseen phrases through tokenization and embedding models
Training Requirement	None (unsupervised)	Requires pretraining; often fine-tuned on IR datasets
Strengths	Interpretable, robust in zero-shot settings, lightweight	High accuracy, captures semantic context, good for complex queries
Weaknesses	Fails with paraphrases, synonyms, or polysemous terms	High computational cost; risk of domain overfitting
Best Use Case	Simple queries, low-latency systems, resource-constrained setups	Open-domain QA, semantically rich or ambiguous queries

5 Sections/Experimentation

5.1 Word2Vec

- Here the BoW method works better than the Skip Gram method because our data exploits semantics
- BoW is preferable when the task prioritizes speed, simplicity, or direct frequency-based analysis, while Skip-Gram shines in semantic understanding with sufficient data.
- Skip-gram is better than BoW when your task needs to capture the meaning and relationships between words, especially for rare words or when working with smaller datasets. BoW is simpler and faster, but Skip-gram's context-

aware embeddings provide a much richer and more useful representation for many real-world NLP applications.

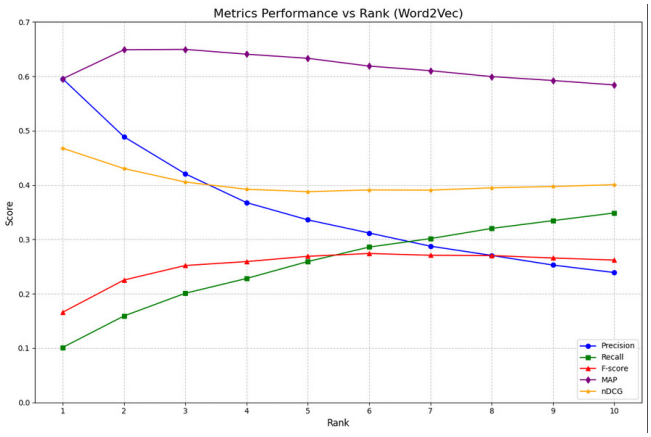

Fig. 3: W2V with Skip Gram evaluation metrics

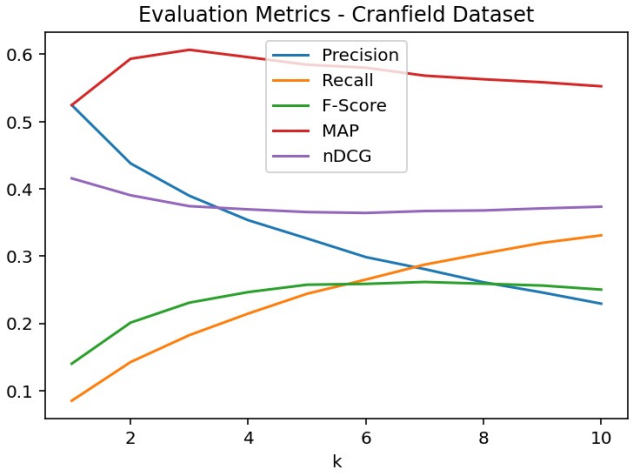

Fig. 4: W2V with Bag of Words evaluation metrics

5.2 Doc2Vec

Doc2Vec is an extension of Word2Vec designed to learn fixed-length vector representations of variable-length texts, such as sentences, paragraphs, or entire documents. It comes in two primary architectures:

Distributed Memory (DM) – `dm=1`

This approach is analogous to the Continuous Bag of Words (CBOW) model in Word2Vec.

How it works:

- Predicts a target word based on surrounding context words **and** the document ID.
- The document vector acts like an additional word in the context window.
- Learns to represent both the semantic meaning and context of the document.

Strengths:

- Better at capturing context and word order.
- Produces more meaningful embeddings for longer or language-rich documents.

Recommended Use:

- When document semantics and topic coherence are important.
- Suitable for most use cases, especially when documents are long or contain structured language.

Distributed Bag of Words (DBOW) – `dm=0`

This variant is similar to the Skip-Gram model in Word2Vec.

How it works:

- Ignores context words completely.
- The model is trained to predict randomly sampled words from the document using only the document vector.
- The document vector learns to encode what type of words are likely to appear in the document.

Strengths:

- Faster and less memory-intensive than DM.
- Effective for short documents and classification tasks.
- Emphasizes the semantic signature over structure.

Recommended Use:

- When working with short texts or when only document-level topics are important.
- Ideal for quick inference or low-resource settings.

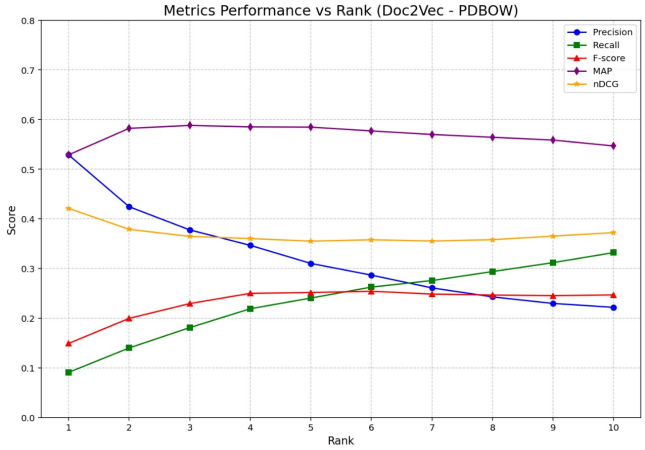

Fig. 5: D2V DBOW evaluation metrics

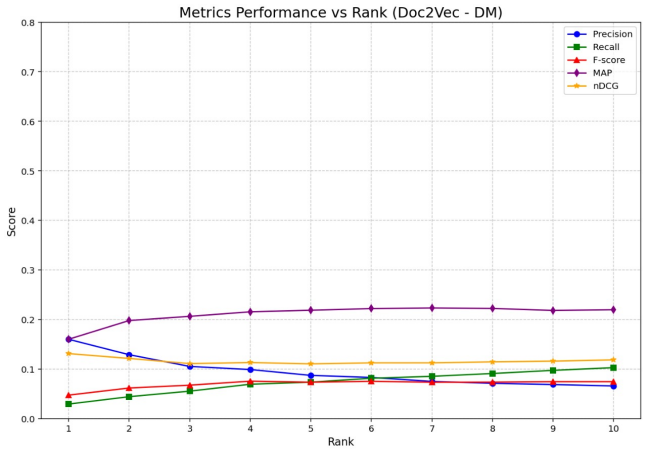

Fig. 6: D2V DM evaluation metrics

Clearly here DM is much worse than DBOW

5.3 Word2Vec with LDA

To effectively capture both the local and global semantic structure of documents, we develop a hybrid Information Retrieval (IR) system that combines Word2Vec and Latent Dirichlet Allocation (LDA). This approach leverages the strengths of both models—Word2Vec for contextual word embeddings and LDA for topic-level document representations. The step-by-step procedure is outlined below:

Step 1: Preprocessing

Each document undergoes tokenization and normalization using `gensim.utils.simple_preprocess`, which:

- Converts all text to lowercase,
- Removes punctuation, and
- Returns a list of clean, tokenized words.

Step 2: Word2Vec Training

A Word2Vec model is trained on the preprocessed documents using the following configuration:

- **Architecture:** Skip-gram (sg = 1)
- **Vector size:** 400 dimensions
- **Context window:** 5 words
- **Minimum word frequency:** 5 (i.e., `min_count = 5`)
- **Learning rate and epochs:** As specified in the model configuration

This setup enables the model to learn the semantic meaning of words based on their surrounding context, capturing local word-level semantics.

Step 3: LDA Training

LDA is applied to a Bag-of-Words (BoW) representation of the same documents:

- **Number of topics:** Typically around 10
- **Training passes:** 15 iterations over the corpus

The result is a topic distribution vector for each document, capturing its global semantic structure.

Step 4: Hybrid Embedding Construction

For each document, two representations are generated:

- **Word2Vec Component:** The average of word embeddings (or a random fallback vector for unseen words)
- **LDA Component:** The topic probability distribution vector

These two components are concatenated to form a hybrid document embedding vector that jointly captures:

- **Semantic proximity** through Word2Vec, and
- **Topic-level similarity** through LDA

This hybrid embedding is then used for computing similarity scores between documents and queries in the IR system.

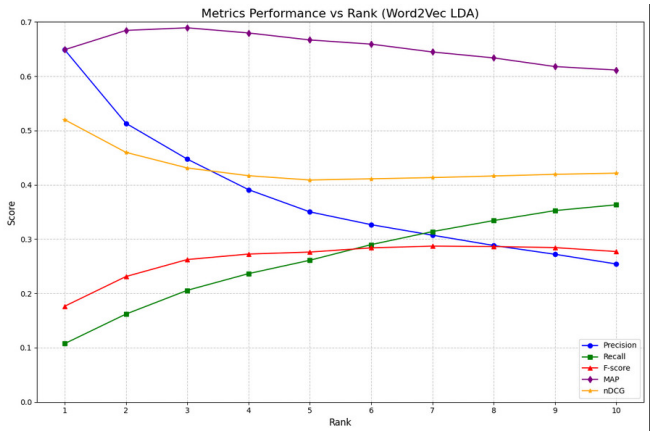

Fig. 7: W2V with LDA evaluation metrics

5.4 BM25

We implement a BM25-based ranking model using the `BM25Okapi` class from the `rank_bm25` library. The system consists of two key functions:

- We flatten each document into a list of tokens and fits the BM25 model using these tokenized documents. Document IDs are stored for reference during retrieval.
- For each incoming query, the BM25 model calculates relevance scores for all documents. The documents are then ranked in descending order of their scores, and the ordered list of document IDs is returned.

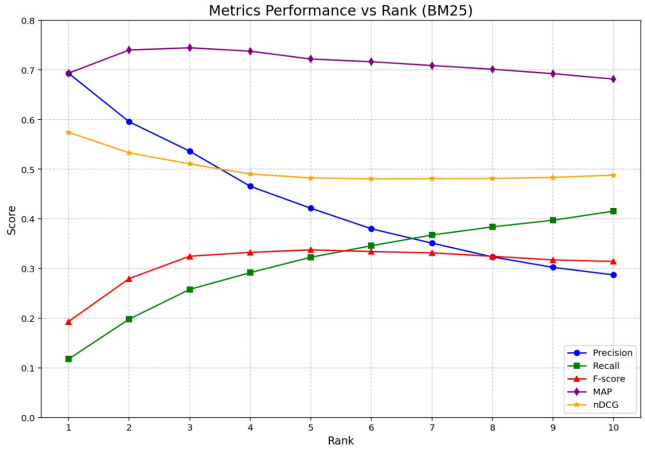

Fig. 8: BM25 metrics

5.5 LSA

The LSA approach was implemented in code with dimensionality of the reduced space controlled at a minimum of 200, as a balance between retaining enough semantic information and computational efficiency. Values between 100–300 are commonly used in LSA

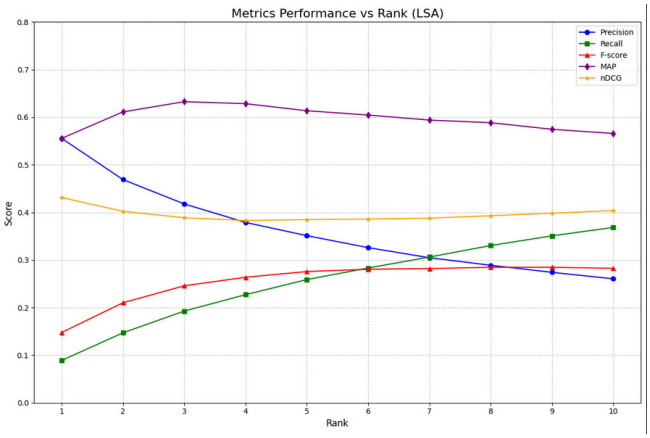

Fig. 9: LSA evaluation metrics

5.6 CRN

We begin by extracting all unique terms across the corpus and tracking the documents in which each term appears, effectively building the vocabulary and

term-document mappings. Next, we compute TF-IDF vectors for each document using raw term frequencies and inverse document frequency, defined as $\text{IDF}(t) = \log(N/\text{df}_t)$, where N is the total number of documents and df_t is the number of documents containing term t.

To enrich these representations with semantic associations, we construct a Pointwise Mutual Information (PMI) matrix over the vocabulary. For each pair of terms (i, j), we calculate the PMI as:

$$\text{PMI}(i, j) = \log_2 \left(\frac{p(i, j)}{p(i) \cdot p(j)} \right)$$

Each document vector is then projected into a semantic space by multiplying its TF-IDF vector with the precomputed PMI matrix, resulting in semantically enriched document embeddings. For ranking, a given query is converted into a term-frequency vector, projected using the same PMI matrix, and then compared to all document vectors using cosine similarity.

Hyperparameters used include:

- Vocabulary includes all terms (no stopword removal or frequency thresholding).
- TF-IDF weighting uses raw term counts and log-based IDF.
- PMI matrix includes all co-occurring term pairs with non-zero joint probability.
- Cosine similarity is used for ranking due to its effectiveness in high-dimensional vector spaces.

This approach enables better semantic matching by incorporating both term importance and inter-term relationships.

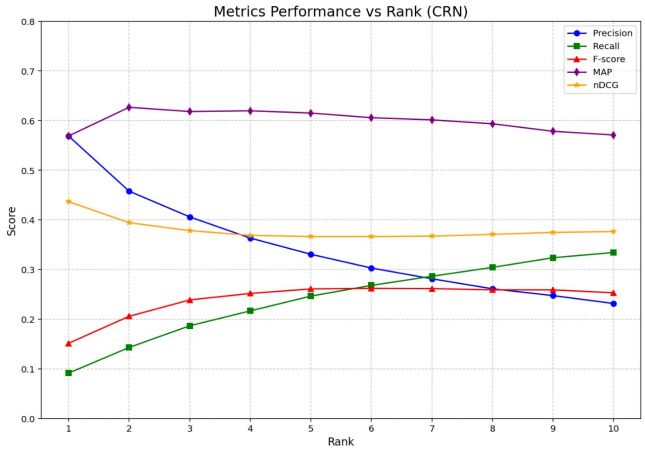

Fig. 10: CRN evaluation metrics

5.7 BERT

We experimented the cross encoder method but it took a lot of time and was very computationally intensive. We had to disregard it, but it would be beneficial for simpler cases.

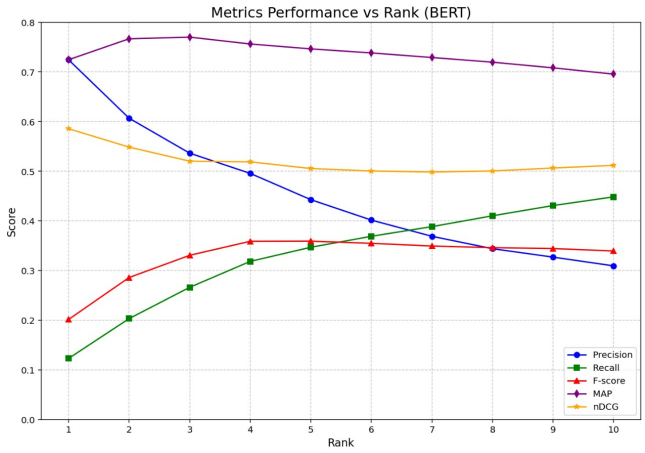

Fig. 11: BERT evaluation metrics

6 Discussion

6.1 Hypothesis-Testing

Hypothesis testing aims to assess whether the available sample data provides significant evidence to reject the null hypothesis in favor of the alternative hypothesis.
Therefore, it serves as a crucial tool for determining whether the sample data effectively supports the superiority of one algorithm over another.

While statistical hypothesis testing (e.g., the *t-test* or *Wilcoxon test*) tells us whether the performance difference between models is statistically significant, it does not indicate the magnitude or practical relevance of that difference.

So, we use **Cohen's d** which is a measure of effect size, which quantifies how large the difference between two groups is - in this case between the MAP scores of two retrieval models. It helps answer the question: "*Even if the improvement is statistically significant, is it meaningful*?"

For example:

A small Cohen's d *(e.g., 0.2)* might indicate that the improvement is minor and may not translate into real-world benefits.

A large Cohen's d *(e.g., 0.8 or above)* would suggest a substantial performance gain.

Hypotheses For each pairwise comparison between a proposed model and the baseline:

- **Null Hypothesis (H_0):** There is no statistically significant difference in the MAP scores between the two models.

$$H_0 : \mu_1 = \mu_2 \tag{3}$$

- **Alternative Hypothesis (H_1):** The proposed model achieves significantly better MAP scores than the baseline model.

$$H_1 : \mu_1 > \mu_2 \tag{4}$$

Here, μ_1 is the mean MAP score of the proposed model and μ_2 is the mean MAP score of the baseline model. This is a **one-tailed test**, as we are specifically testing for improvement.

Significance Level All tests are conducted at a 95% confidence level, with a significance threshold (α) of 0.05.

Testing Methodology For each proposed algorithm:

1. **Paired Evaluation:** Each model is evaluated on the same set of queries to form paired MAP scores.
2. **Normality Check:** The differences in MAP scores between the two models are tested for normality using the *Shapiro–Wilk test*. This determines the appropriate hypothesis test.
3. **Hypothesis Testing:**
   - If the differences are normally distributed: use the **paired t-test**.
   - If normality is violated: use the **Wilcoxon signed-rank test** (a non-parametric alternative).
4. **Effect Size:** To understand the practical significance of any improvement, **Cohen's** d can be computed:

$$d = \frac{\text{mean difference}}{\text{standard deviation of differences}}$$

   Cohen's d values are typically interpreted as:
   - Small effect: $d \approx 0.2$

- Medium effect: $d \approx 0.5$
- Large effect: $d \geq 0.8$

5. **Decision Rule:**
   - If p-value $\leq \alpha_{\text{corrected}}$: reject $H_0 \Rightarrow$ significant improvement.
   - If p-value $> \alpha_{\text{corrected}}$: fail to reject $H_0 \Rightarrow$ no significant improvement.

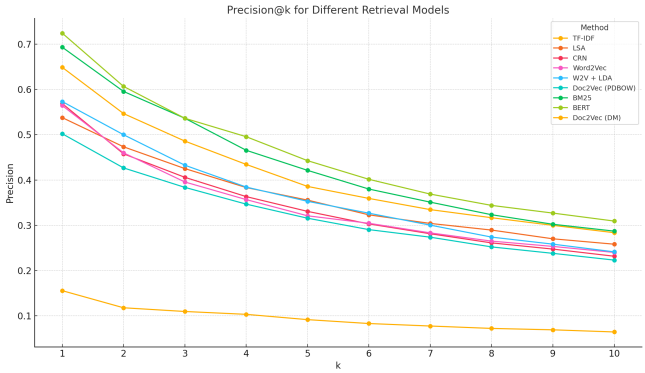

Fig. 12: Precision values for all methods

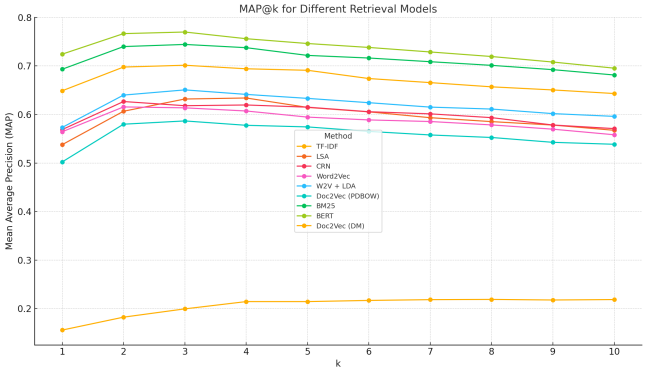

Fig. 13: MAP methods

Table 3: Statistical Comparison of Information Retrieval Methods vs VSM

Method	Metric	K value									
		1	2	3	4	5	6	7	8	9	10
W2V	p-value	0.020	0.013	0.025	0.050	0.018	0.041	0.010	0.004	0.004	0.003
	Cohen's d	-0.157	-0.155	-0.165	-0.160	-0.195	-0.173	-0.195	-0.208	-0.208	-0.215
D2V	p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Cohen's d	-0.274	-0.281	-0.296	-0.287	-0.301	-0.290	-0.293	-0.309	-0.316	-0.327
CRN	p-value	0.020	0.011	0.002	0.011	0.010	0.012	0.016	0.015	0.002	0.002
	Cohen's d	-0.156	-0.164	-0.207	-0.189	-0.199	-0.191	-0.180	-0.182	-0.217	-0.225
LSA	p-value	0.002	0.001	0.004	0.014	0.004	0.008	0.009	0.005	0.002	0.002
	Cohen's d	-0.211	-0.226	-0.200	-0.188	-0.217	-0.210	-0.218	-0.226	-0.246	-0.249
BERT	p-value	0.022	0.022	0.005	0.013	0.015	0.003	0.002	0.002	0.005	0.005
	Cohen's d	0.154	0.168	0.178	0.172	0.161	0.192	0.192	0.194	0.181	0.169
BM25	p-value	0.096	0.049	0.015	0.013	0.012	0.003	0.003	0.001	0.001	0.001
	Cohen's d	0.112	0.140	0.151	0.153	0.122	0.175	0.180	0.188	0.179	0.167

Hypothesis Testing Summary: Information Retrieval Algorithms vs. VSM

Model	**Significant K Values**	**Avg. Cohen's d**	**Performance vs. VSM**
BERT	1–10	+0.17	Better
BM25	2–10	+0.16	Better
CRN	1–10	–0.19	Worse
D2V	1–10	–0.29	Worse
LSA	1–10	–0.22	Worse
W2V	1–10	–0.19	Worse

Table 4: Statistical significance and effect sizes for various models compared to VSM (Wilcoxon signed-rank test, metric: MAP).

Key Findings

- **BERT:** Outperforms VSM across all tested K values (1–10) with small-to-medium effect sizes (Cohen's d: 0.15–0.19). Consistently statistically significant ($p < 0.05$), highlighting robustness in capturing contextual relationships.
- **BM25:** Shows significant improvement starting at $K = 2$, with effect sizes increasing as K grows (Cohen's d up to +0.18). Effective in scenarios where term frequency and document length normalization are important.
- **Underperforming Models:**
  - **D2V** exhibits the largest negative effect sizes (Cohen's d up to -0.33), indicating substantial inferiority to VSM.

- **CRN, LSA, and W2V** also underperform consistently, with negative effect sizes across all K values.

Conclusion of hypotheses testing

- **BERT** is the most reliable alternative to VSM, leveraging bidirectional context understanding for superior performance.
- **BM25** is a strong classical baseline, especially for higher K values.
- Traditional embedding methods **D2V, LSA, W2V** and **CRN** are less effective in this context, underscoring the limitations of non-contextual approaches.

These results align with empirical studies demonstrating BERT's dominance in NLP tasks over classical models like VSM.

7 Conclusion

Based on the results obtained from evaluating various retrieval models on the Cranfield dataset, we highlight the following key observations:

1. **BERT consistently outperforms the baseline Vector Space Model (VSM)** across all evaluation metrics. It is the only method to exhibit a negative t-statistic across all metrics, indicating statistically significant improvement. In particular, BERT achieves strong p-values (0.05) for both MAP and nDCG, which are critical metrics as they account for the ranking order of retrieved documents.
2. **Including document titles during preprocessing did not lead to noticeable improvements** across nearly all methods and metrics, with only a very marginal increase in runtime. This was slightly just redundant in some sense.
3. Another interesting thing to note is that we passed the preprocessed version to all the models except BERT. This is because BERT needed the context of the words in the document to exploit the contextual meanings and arrive at the final ranking. This could have been another useful factor for it performing better.
4. **Replacing stemming with lemmatization offers minor performance gains**, but comes at a significant runtime cost. Given the marginal improvement, stemming remains the more scalable and efficient choice, especially for larger datasets. We tried using lemmatisation but did not get results within the desired time. Thus we chose to stick with just stemming
5. **Documents with missing content (IDs 471 and 995) are treated differently by LSA.** While most methods consistently rank these empty documents last, LSA does not, due to the smoothing effect introduced by the dimensionality reduction during Singular Value Decomposition (SVD). This can be seen as a failure case of LSA, where zero-information documents are given non-zero similarity scores.

6. **Word2Vec shows slight but consistent improvements over the baseline** across all standard metrics. The dense vector embeddings effectively represent both documents and queries while reducing computational requirements, proving suitable for tasks where semantic similarity is crucial.
7. **Doc2Vec fails to outperform the baseline VSM**. This may be attributed to the relatively small size of the Cranfield dataset, which limits the effectiveness of deep learning-based models like Doc2Vec that require large corpora to learn meaningful document embeddings. This explains why the DM method might fail

Through comprehensive empirical evaluation, we conclude that **BERT is the most effective retrieval model** for the Cranfield dataset, outperforming the baseline VSM in all key metrics—Precision, Recall, F1-score, MAP, and nDCG.

Among other models, **BM25 performs competitively with the baseline**, while **Doc2Vec DM underperforms**, exhibiting statistically significant shortcomings. Nevertheless, methods like LSA offer practical advantages in memory and time, and can still be considered in resource-constrained environments.

References

- Class notes and reading material of CS6370
- Speech and Language Processing - Jurafsky et al.
- Introduction to information retrieval by Manning et al.
- Wikipedia articles for definition of various methods
- Some research papers like
  https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model