Bridging Lexical and Semantic Gaps in Information Retrieval: A Comparative Study on the Cranfield Dataset

https://github.com/Jaanav-Mathavan/NLP.git

Indian Institute of Technology, Madras, Chennai 600036, TN, India
ee22b002,ee22b167,be21b034,bs21b032,ee22b024@smail.iitm.ac.in

Abstract. In this project report, we investigate the retrieval effectiveness of a range of Information Retrieval models on the Cranfield dataset. Starting from the classical Vector Space Model (TF–IDF) and its probabilistic extension BM25, we implement and compare semantic and hybrid methods including Latent Semantic Analysis (LSA), Explicit Semantic Analysis (ESA), Normalized ESA (NESA), WordNet–based query expansion, and dense transformer embeddings (All-MiniLM). A modular preprocessing pipeline (sentence segmentation, tokenization, stemming, stopword removal) feeds into each retrieval variant. We evaluate each model using MAP@10, Precision@10, Recall@10, and nDCG@10, and apply paired statistical tests on per-query AP@10 and nDCG@10 scores to assess significance under an assumption of independent, approximately normal per-query differences. Our results demonstrate that BM25 consistently outperforms other methods, while TF–IDF remains a competitive baseline; semantic enhancements improve recall at the cost of precision, highlighting trade-offs that future domain-adaptive and hybrid weighting strategies could address.

Keywords: Information Retrieval, Cranfield dataset, Vector Space Models, TF-IDF, Explicit Semantic Analysis(ESA), WordNet, Latent Semantic Analysis(LSA), Autocomplete

1 Introduction

Search engines underpin the modern information landscape, yet the traditional Vector-Space Model (VSM) based on TF–IDF weights often struggles with semantic gaps, query mismatches and user input errors. In technical domains such as aerospace engineering, exact term overlaps may be sparse, leading to degraded retrieval performance and suboptimal user experiences.This project aims to overcome these limitations by exploring a suite of complementary methods:

- **Latent Semantic Analysis (LSA)** to reduce dimensionality and capture hidden conceptual structure.
- **Explicit Semantic Analysis (ESA)** and **Normalized ESA (NESA)** to map texts into a Wikipedia-derived concept space.
- **WordNet-based expansion** to enrich term representations with lexical semantic relations.
- **Dense embeddings** using a pre-trained MiniLM transformer to encode contextual semantics.
- **Query autocomplete** via n-gram models and trie structures to handle incomplete or misspelt queries.

We evaluate each approach against the TF–IDF baseline on the Cranfield dataset of 20 queries using MAP@10, Precision@10, Recall@10 and nDCG@10. To rigorously validate improvements, we formulate paired t-test hypotheses comparing TF–IDF against each alternative model under the assumption that per-query average precisions are independent and approximately normally distributed. The findings provide insights into which methods yield statistically significant gains and guide hybrid model design for robust, semantic-aware retrieval.

2 Problem Statement

The performance of traditional Information Retrieval (IR) systems, based on the Vector Space Model (VSM), is often limited by factors such as semantic gaps, spelling errors, and query mismatches. These limitations hinder the effectiveness and efficiency of the retrieval process, leading to suboptimal user experiences. This project focuses on improving the VSM for information retrieval, with particular attention to enhancing retrieval efficiency, increasing robustness to spelling errors, and improving query autocomplete functionality.

Methods such as Latent Semantic Analysis (LSA) and Explicit Semantic Analysis (ESA) are supposed to be implemented to address these issues. Through rigorous hypothesis testing, this project aims to demonstrate how these methods can significantly improve the query suggestion process and overall search engine performance.

3 Preprocessing (Warm-up)

3.1 Sentence Segmentation

Sentence segmentation is a fundamental preprocessing step in information retrieval that involves breaking down a document into individual sentences.

We implemented two approaches to achieve this:

- **Top-Down approach:** It uses punctuation cues while carefully handling abbreviations and lists to avoid incorrect splits. It identifies sentence boundaries based on terminating punctuation (., ?, !, ;) but checks for known exceptions (e.g., "Dr.", "e.g.", "1.") to prevent incorrect segmentation.
- **Bottom-Up approach:** Punkt sentence tokeniser from NLTK is used, which has been trained on a large corpus to infer sentence boundaries.

Example:

- **Document:** *experimental investigation of the aerodynamics of a wing in a slipstream . an experimental study of a wing in a propeller slipstream was made in order to determine the spanwise distribution of the lift increase due to slipstream at different angles of attack of the wing and at different free stream to slipstream velocity ratios .*
- **After Sentence Segmentation** [*"experimental investigation of the aerodynamics of a wing in a slipstream ."*, *"an experimental study of a wing in a propeller slipstream was made in order to determine the spanwise distribution of the lift increase due to slipstream at different angles of attack of the wing and at different free stream to slipstream velocity ratios ."*]

3.2 Tokenisation

Tokenisation is a preprocessing step that breaks sentences into individual words or tokens. We first convert all tokens to lowercase for consistency and prepare the text for further preprocessing steps.

We implemented two methods for this:

- **Naive Regex-based tokeniser:** The naive approach extracts only alphanumeric words using regular expressions, removing punctuation, symbols, and partial word fragments.
- **Penn Treebank Tokeniser:** The Penn Treebank method uses a rule-based tokeniser from NLTK and filters out non-alphanumeric tokens to achieve similar results.

Examples:

- **Sentence:** *what problems of heat conduction in composite slabs have been solved so far .*
- **Tokenized sentence:** [*"what"*, *"problems"*, *"of"*, *"heat"*, *"conduction"*, *"in"*, *"composite"*, *"slabs"*, *"have"*, *"been"*, *"solved"*, *"so"*, *"far"*, *"."*]

3.3 Inflection Reduction (Stemming)

Inflection reduction is a key preprocessing step that simplifies words to their base form to reduce redundancy and dimensionality. We use **Porter's stemming algorithm** to implement this on each token in the text. Stemming works by stripping common suffixes ("*running*" → "*run*", "*easily*" → "*eas*"), without necessarily ensuring the result is a valid word. It is a fast and efficient method, especially useful in information retrieval tasks, where exact grammatical correctness is less critical. Compared to lemmatisation, stemming is less linguistically accurate but computationally cheaper.

Examples:

- **Tokens:** *Experimental, Investigation, aerodynamics*
- **After inflection reduction:** *Experiment, Investig, aerodynam*

3.4 Stopword Removal

Stopword removal is a preprocessing step that filters out commonly used words (like *"the", "is", "and"*) which don't provide any discrimination between documents in an information retrieval context. We use **NLTK's built-in list of English stopwords** and process the input by retaining only meaningful, alphanumeric tokens that are not in the stopword list.

Another way of removing stopwords is by using the **Inverse Document Frequency (IDF) scores**. The IDF scores of the terms represent their discriminating power in the corpus. Hence, this method has the advantage of removing domain-specific stopwords having an IDF score less than a threshold value. We have not used this method in our models because the effectiveness of removal depends heavily on the threshold value itself.

4 Query Autocomplete

Query autocomplete is an additional feature used in Information retrieval (IR) to make the system robust to incomplete and erroneous queries. It assists users by predicting and completing their queries based on partial input. We include a similar autocomplete functionality in our toy IR system. Users can input an **incomplete or misspelled query**, and the system will automatically complete the rest based on the most similar complete query in the corpus.

Misspelled Queries:

For misspelled queries, traditional autocomplete models often fail because they cannot detect word patterns in the training corpus. We make our query robust to spelling errors by iteratively checking if each word in the query is present in the vocabulary and if not match it to the most similar word. This makes our IR system **robust to spelling errors**. We implement this using the `difflib` module.

Assumption: *The incomplete user query will always be a proper subset of any one or more queries present in our corpus.*

We model this using two approaches:

4.1 N-gram Model

The N-gram model predicts the next word in a query by analysing the frequency of word sequences ('n-grams') in a training corpus. Given a prefix, it identifies the most likely next word based on past occurrences of similar word patterns, enabling query autocompletion one word at a time. We keep on iterating this until the input query matches with one of the queries in the corpus.

Observations: For N=2 the N-gram model performs very poorly as it is unable to find a completed query due to the existence of stop words which occur very frequently. Thus when it tries to match the query based on the word which has the highest probability to come next it gets stuck in an infinite loop where it never matches with any query. Setting higher values of N like N=3 helps resolve this issue.

User Query 1: *des a praccal fw*
Autocompleted Query 1: *does a practical flow follow the theoretical heat transfer to a blunt body*

User Query 2: *what chemical kinetic*
Autocompleted Query 2: *what chemical kinetic system is applicable to hypersonic aerodynamic problems.*

Perplexity Perplexity is an intrinsic evaluation metric that quantifies how well an N-gram language model predicts a sequence of words. In query autocompletion, it is used to evaluate the quality of the autocompleted portion of a query—that is, the model's confidence in the words it appends after the original user input (after spelling correction). Let the original query contain m tokens and the autocompleted query contain T tokens. Then, perplexity is computed only on the suffix starting from the $(m+1)^{\text{th}}$ token, using the N-gram context that includes up to $n-1$ tokens from the original query. The formula is:

$$\log(\text{Perplexity}) = -\frac{1}{T-m-n+1} \sum_{i=m}^{T-n} \log(P(w_{i+n-1} \mid w_i, \ldots, w_{i+n-2})) \tag{1}$$

where $P(w_{i+n-1} \mid w_i, \ldots, w_{i+n-2})$ is the probability of the next word given the N-gram prefix. Lower perplexity indicates that the model assigns higher likelihood to the generated sequence, implying more fluent and accurate autocompletions.

User Query: cn te
Autocompleted Query: can the aerodynamic performance of channel flow ground effect machines be calculated .
Perplexity: 513.5889354156924

4.2 Tries

In this method, we employ the Trie data structure to store the corpus queries and when an incomplete query is encountered, we search the Trie to identify all possible completions that begin with the given prefix. The earliest matching complete query or the most frequent one (based on corpus statistics) is then suggested as the auto-completion result.

Observations: The tries model for autocompleting queries does not get stuck in any infinite loops and is more robust to stop words. But unlike the N-gram model it cant be extended to autocomplete queries which are not present in the corpus.

User Query 1: *what is the magnitude and dist*
Autocompleted Query 1: *what is the magnitude and distribution of lift over the cone and the cylindrical portion of a cone-cylinder configuration .*

User Query 2: *what chemical kinetic*
Autocompleted Query 2: *what chemical kinetic system is applicable to hypersonic aerodynamic problems.*

User Query 3: *des a praccal fw*
Autocompleted Query 3: *does a practical flow follow the theoretical concepts for the interaction between adjacent blade rows of a supersonic cascade .*

Interactive Autocomplete To improve the performance of our query autocompleting, an interactive interface was implemented, which, given a user-input query, gave the top k possible autocomplete options from which the user can choose, which is the option most appropriate for them, similar to the Google Search website suggestions. To generate the suggestions, we use the Tries models we had used earlier, with minor modifications to allow generation of the top k suggestions.

User Query: *cn te*
Autocomplete Suggestions:

1. *can the transverse potential flow about a body of revolution be calculated efficiently by an electronic computer.*
2. *can the transonic flow around an arbitrary smooth thin airfoil be analysed in a simple approximate way.*

3. *can the three-dimensional problem of a transverse potential flow about a body of revolution be reduced to a two-dimensional problem.*
4. *can the three-point boundary-value problem for the Blasius equation be integrated numerically, using suitable transformations, without iteration on the boundary conditions.*
5. *can the procedure of matching inner and outer solutions for a viscous flow problem be applied when the main stream is a shear flow.*

Selected Autocompleted Query: *can the transonic flow around an arbitrary smooth thin airfoil be analysed in a simple approximate way.*

5 Information Retrieval

This section introduces the information retrieval methods implemented to rank documents based on query relevance, evaluated using the Cranfield dataset of aerospace engineering texts. The implemented methods include Term Frequency-Inverse Document Frequency (TF-IDF), Query and Document Expansion with WordNet, WordNet Similarity Matrix, Latent Semantic Analysis (LSA), Sentence Transformers, BM25, Explicit Semantic Analysis (ESA), Normalized Explicit Semantic Analysis (NESA), and a hybrid approach combining TF-IDF with ESA. These methods generate document and query representations to facilitate similarity-based ranking within a unified retrieval framework.

5.1 Term Frequency-Inverse Document Frequency

The Term Frequency-Inverse Document Frequency (TF-IDF) model forms the cornerstone of the information retrieval framework developed for ranking documents in the Cranfield dataset, which comprises aerospace engineering texts. The intuition behind TF-IDF is to assign weights to terms based on their significance within a document relative to their prevalence across the entire corpus. Terms that appear frequently in a specific document but rarely in others are deemed more discriminative, thus receiving higher weights. This approach enables effective retrieval by prioritizing documents containing terms closely aligned with the query, particularly suited for the technical and domain-specific terminology encountered in the Cranfield dataset.

In this work, the TF-IDF model was implemented to construct a document-term matrix, where each document is represented as a vector of term weights. The method incorporates two-word phrases to capture limited contextual relationships, enhancing the representation of aerospace-related concepts. Queries are similarly transformed into vector representations, and documents are ranked according to their similarity to the query, leveraging the vector-space model to quantify relevance. This implementation provides a robust baseline for term-based retrieval, effectively handling exact term matches prevalent in the dataset's query-document pairs.

The mathematical formulation of TF-IDF is as follows. For a term t in document d within a corpus D, the TF-IDF weight is computed as:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \cdot \text{IDF}(t, D), \tag{2}$$

where:

- $\text{TF}(t, d) = \frac{\text{count}(t,d)}{\sum_{t' \in d} \text{count}(t',d)}$ represents the term frequency, defined as the normalized count of term t in document d, with $\text{count}(t, d)$ denoting the number of occurrences of t.
- $\text{IDF}(t, D) = \log\left(\frac{|D|}{|\{d \in D : t \in d\}|+1}\right)$ is the inverse document frequency, where $|D|$ is the total number of documents, $|\{d \in D : t \in d\}|$ is the number of documents containing t, and the $+1$ term prevents division by zero.

The document-term matrix $X \in \mathbb{R}^{|D| \times |V|}$, where V is the vocabulary, has entries $X_{d,t} = \text{TF-IDF}(t, d, D)$. For a query q, a TF-IDF vector is computed analogously, and the relevance of document d is determined by the cosine similarity:

$$\text{cosine similarity}(q, d) = \frac{q \cdot d}{\|q\|_2 \|d\|_2}, \tag{3}$$

where $\cdot$ denotes the dot product, and $\|\cdot\|_2$ is the Euclidean norm. This formulation ensures that documents with higher similarity scores are ranked more favorably, aligning with the retrieval objectives for the Cranfield dataset.When we considered bigrams along with unigrams the perormanc significantly increased

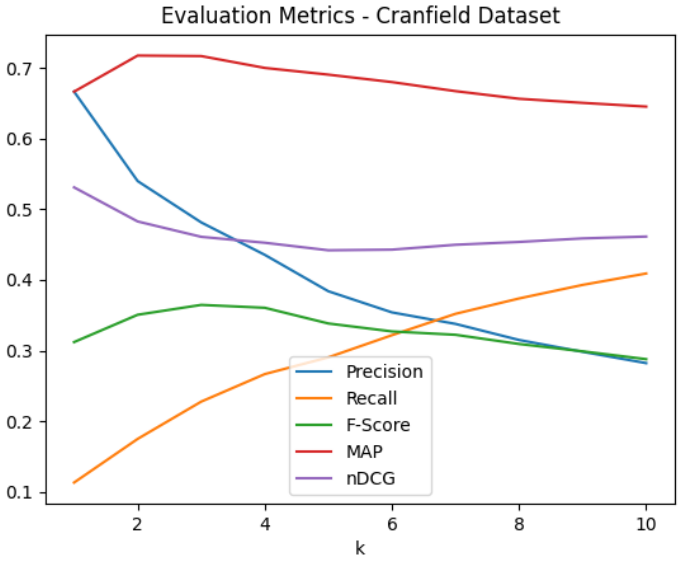


Fig. 1: Comparison of IR metrics (MAP, Precision, Recall, nDCG, $F_{0.5}$ score.) across document ranks using TF-IDF retrieval model

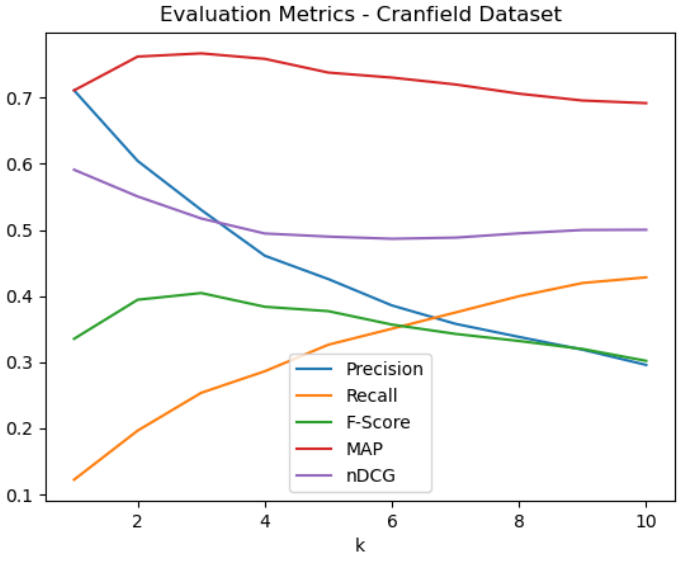


Fig. 2: Comparison of IR metrics (MAP, Precision, Recall, nDCG, $F_{0.5}$ score.) across document ranks using TF-IDF with bigrams retrieval model

Limitations and Proposed Solutions for the Vector Space Model

Despite its usefulness, the Vector Space Model (VSM) exhibits several key limitations. The following approaches were adopted to mitigate these issues:

- **High Computational Cost:** Recomputing document vectors is expensive, especially when new terms are introduced. *To address this, optimized libraries such as scikit-learn were used, and redundant recomputations were avoided.*
- **Loss of Word Order:** As a bag-of-words model, VSM ignores the order of terms, leading to semantically indistinct results for differently phrased queries. *This was mitigated by incorporating semantic models like ESA/NESA and dense embeddings that capture contextual meaning.*
- **Inefficiency with Long Documents:** VSM struggles with accurately computing similarity for lengthy documents. *BM25 was adopted to handle this issue, as it includes document length normalization.*
- **False Negatives Due to Synonymy:** Different words with similar meanings (synonyms) are not recognized as related, leading to missed relevant documents. *Query expansion using WordNet synsets was implemented to include semantically related terms.*
- **False Positives from Lexical Preprocessing:** Errors due to aggressive stemming and tokenization may match irrelevant documents. *This was addressed by incorporating LSA and ESA, which enable concept-based rather than purely lexical matching.*
- **Limited Semantic Representation:** VSM does not capture deeper semantic context or conceptual understanding. *Dense embeddings using Sentence Transformers were used to better represent and compare documents on a semantic level.*

5.2 Latent Semantic Analysis

Latent Semantic Analysis (LSA) improves information retrieval by capturing hidden semantic relationships between terms and documents through dimensionality reduction. Applied to the Cranfield dataset, LSA addresses vocabulary mismatches by mapping documents and queries into a lower-dimensional latent space where semantically related terms are grouped. LSA uses Singular Value Decomposition (SVD) on the document-term matrix $X \in \mathbb{R}^{|D| \times |V|}$ to approximate:

$$X \approx U_k \Sigma_k V_k^T, \tag{4}$$

where U_k, Σ_k, and V_k represent the top k components of documents, singular values, and terms, respectively. Documents are represented as:

$$X_{\text{LSA}} = U_k \Sigma_k, \tag{5}$$

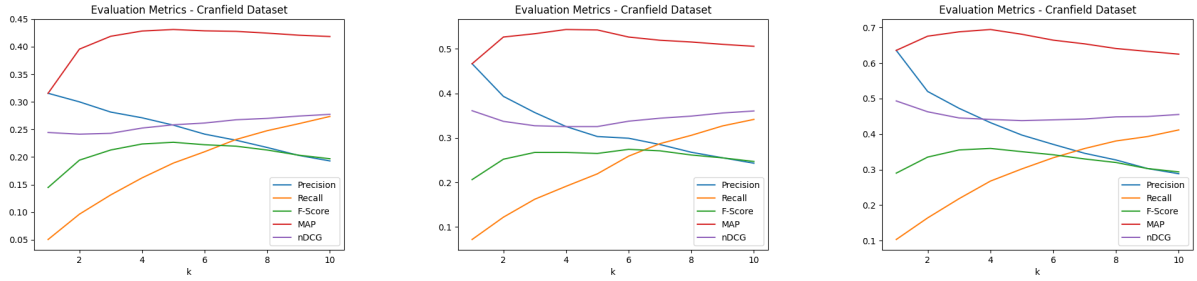
and a query vector $q \in \mathbb{R}^{1 \times |V|}$ is transformed as:

$$q_{\text{LSA}} = q V_k \Sigma_k^{-1}. \tag{6}$$

Similarity is computed in the latent space using cosine similarity:

$$\text{cosine similarity}(q_{\text{LSA}}, d_{\text{LSA}}) = \frac{q_{\text{LSA}} \cdot d_{\text{LSA}}}{\|q_{\text{LSA}}\|_2 \|d_{\text{LSA}}\|_2}. \tag{7}$$

This method enhances retrieval by aligning documents with queries based on underlying concepts, making it effective for the specialized vocabulary of the Cranfield dataset.



(a) LSA with 20 factors (b) LSA with 50 factors (c) LSA with 150 factors

Fig. 3: Comparison of IR metrics (MAP, Precision, Recall, nDCG, $F_{0.5}$ score) across document ranks using LSA models with different number of factors.

5.3 BM25 Ranking Function

BM25 (Best Matching 25) is a probabilistic retrieval model that scores documents based on the presence and frequency of query terms, adjusted by document length and term distribution across the corpus. It is a refinement of the TF-IDF scheme and belongs to the family of bag-of-words ranking functions.Given a query $Q = \{q_1, q_2, \ldots, q_n\}$ and a document D, the BM25 score is computed as:

$$\text{BM25}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)} \tag{8}$$

where:

- $f(q_i, D)$ is the frequency of term q_i in document D
- $|D|$ is the length (in terms) of document D
- avgdl is the average document length in the corpus
- k_1 and b are tunable hyperparameters, typically $k_1 = 1.5$ and $b = 0.75$
- $\text{IDF}(q_i)$ is the inverse document frequency of q_i, defined as:

$$\text{IDF}(q_i) = \log\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right) \tag{9}$$

where:

- N is the total number of documents
- $n(q_i)$ is the number of documents containing the term q_i

The BM25 function accounts for:

1. **Term frequency saturation** — Additional occurrences of a term contribute less to the score.
2. **Document length normalization** — Longer documents are penalized unless justified by proportionally higher term frequencies.
3. **Global term rarity** — Rare terms contribute more to the score than common ones.

This combination makes BM25 highly effective in practical information retrieval systems.

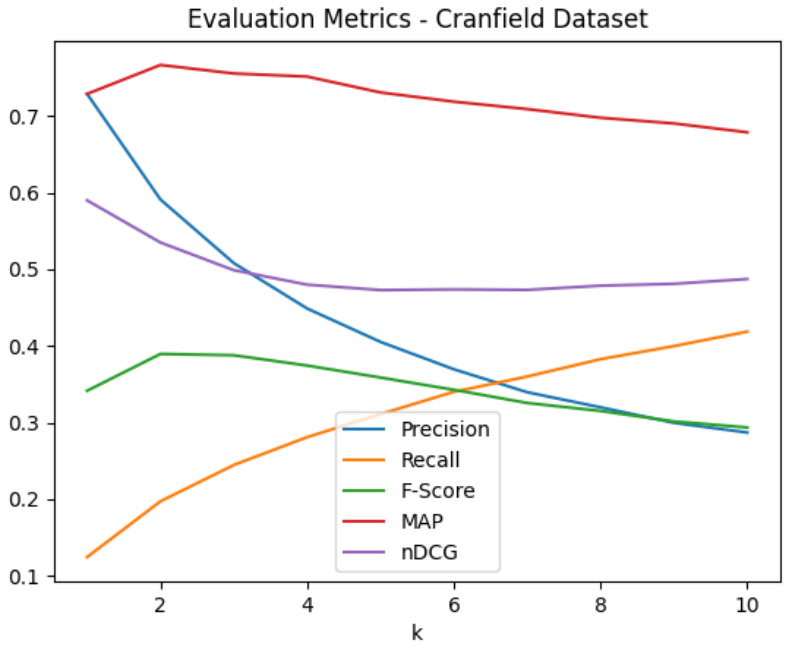


Fig. 4: Comparison of IR metrics (MAP, Precision, Recall, nDCG, $F_{0.5}$ score) across document ranks using BM25 retrieval model

5.4 Query Expansion using WordNet

Query Expansion using WordNet is a semantic augmentation technique employed to improve recall in the information retrieval framework applied to the Cranfield dataset. The method enriches original queries by appending semantically related terms—specifically synonyms—from WordNet, a lexical ontology that organizes words into synsets. This approach is designed to mitigate vocabulary mismatch, a common limitation in keyword-based retrieval systems, by enabling matches between conceptually equivalent terms. For instance, augmenting the query term "aircraft" with its synonym "airplane" allows retrieval of relevant documents that may not contain the original query term explicitly.

In this implementation, each query term is examined for its corresponding synsets in WordNet. Synonyms are then selected from these synsets and added to the query vector, provided they exist in the system's controlled vocabulary. This filtering ensures the expanded terms remain relevant to the domain and avoid introducing noise. The method was integrated into the retrieval pipeline prior to similarity computation, enhancing the ability of the system to retrieve documents with semantically aligned language.

The mathematical formulation modifies the original query vector $q \in \mathbb{R}^{1\times|V|}$, where $|V|$ is the vocabulary size and q_t represents the weight of term t. For each term t in the query, let $S_t \subseteq V$ denote the set of its WordNet synonyms that also exist in the vocabulary. The expanded query vector $q_{\text{exp}} \in \mathbb{R}^{1\times|V|}$ is computed as:

$$q_{\text{exp},t'} = q_t + \sum_{s\in S_t} w_s \cdot \mathbb{1}_{\{s=t'\}}, \tag{10}$$

where $t' \in V$, $\mathbb{1}_{\{s=t'\}}$ is the indicator function that is 1 when synonym s matches term t', and w_s is the weight assigned to synonym s, typically set to a constant value (e.g., 1) or modulated by semantic similarity scores from WordNet.

Once expanded, the query vector q_{exp} is used for retrieval via cosine similarity against each document vector $d \in \mathbb{R}^{1\times|V|}$:

$$\text{cosine similarity}(q_{\text{exp}}, d) = \frac{q_{\text{exp}} \cdot d}{\|q_{\text{exp}}\|_2 \|d\|_2}, \tag{11}$$

where $\cdot$ denotes the dot product and $\|\cdot\|_2$ is the Euclidean norm. This enhanced formulation allows the system to recognize and retrieve semantically relevant documents even when exact term matches are absent.

Empirical results indicated that query expansion using WordNet modestly improved recall but had mixed effects on precision, reflecting a trade-off between semantic breadth and specificity. Nonetheless, it demonstrated the value of ontology-driven enhancements for domain-sensitive information retrieval tasks like those posed by the Cranfield dataset.

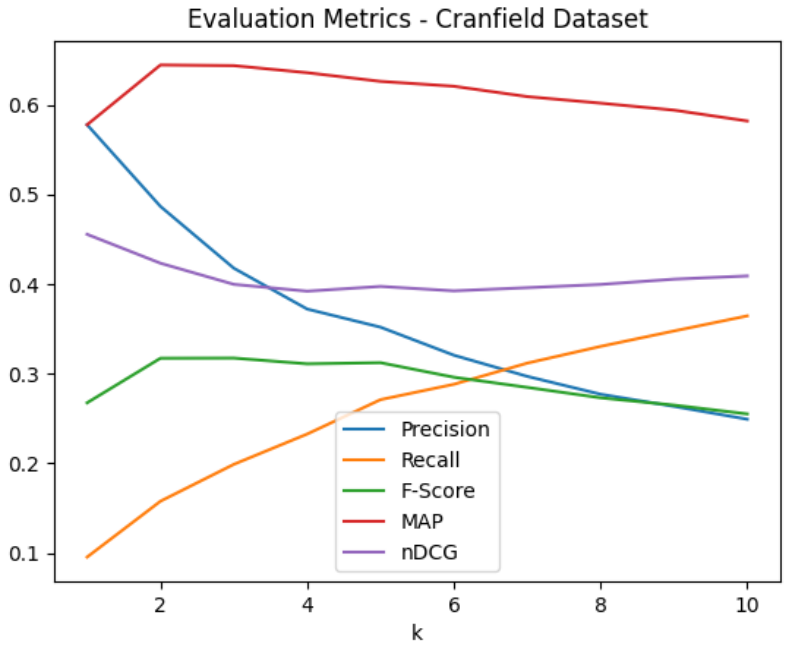


Fig. 5: Comparison of IR metrics (MAP, Precision, Recall, nDCG, $F_{0.5}$ score) across document ranks using TFIDF + Query expansion using Wordnet retrieval model

5.5 WordNet Similarity with Similarity Propagation

This method enhances information retrieval by integrating semantic similarity from WordNet into traditional TF-IDF representations. The central idea is to propagate term importance across semantically related terms, allowing the system to capture meaning even when queries and documents use different but related vocabulary—a common scenario in the Cranfield dataset's technical domain.

Semantic similarity is computed using the Wu-Palmer metric, which quantifies how close two terms are within WordNet's hierarchy based on their shared ancestors. A similarity matrix $S \in \mathbb{R}^{|V|\times|V|}$ is constructed, where $S_{t,t'}$ denotes the Wu-Palmer similarity between vocabulary terms t and t'.

To incorporate this into retrieval, each TF-IDF vector is enriched by propagating term weights through the similarity matrix. For a document vector $X_d \in \mathbb{R}^{1\times|V|}$, the propagated representation is:

$$d_{\text{prop}} = X_d(I + \alpha S), \tag{12}$$

where I is the identity matrix and $\alpha > 0$ controls the strength of semantic propagation. A query vector q is similarly transformed:

$$q_{\text{prop}} = q(I + \alpha S). \tag{13}$$

Retrieval is then performed using cosine similarity between the propagated vectors:

$$\text{cosine similarity}(q_{\text{prop}}, d_{\text{prop}}) = \frac{q_{\text{prop}} \cdot d_{\text{prop}}}{\|q_{\text{prop}}\|_2 \|d_{\text{prop}}\|_2}. \tag{14}$$

While this method theoretically improves semantic matching, its effectiveness on the Cranfield dataset was limited. This is likely due to the mismatch between WordNet's general-purpose vocabulary and the dataset's specialized aerospace terminology. Nonetheless, the approach demonstrates the potential of ontology-based methods for enhancing term-based retrieval with semantic context.

5.6 Wikipedia Dataset Preparation

To construct a semantic concept space for information retrieval over the Cranfield dataset, a domain-specific subset of Wikipedia was curated and processed. Articles were sourced from top-level categories such as aerospace engineering, aeronautics, aircraft, fluid dynamics, and mechanical engineering, including recursive traversal of subcategories to ensure broad and relevant coverage. The dataset initially comprised 5,221 articles and was subsequently expanded to approximately 50,000, from which the top 10,000 were selected as semantic concepts based on document frequency and relevance.

Each article underwent standardized preprocessing: tokenization, lowercasing, punctuation removal, and stopword elimination to reduce noise. Stemming was applied to reduce inflectional variations, improving lexical matching across concept and query spaces. The result was a clean term-document matrix suitable for semantic indexing.

To build the term-to-concept index $C \in \mathbb{R}^{|V|\times|K|}$, each term t in vocabulary V was mapped to concepts $k \in K$ using TF-IDF weighting. The term frequency (TF) for a term t in concept k was computed as:

$$\mathrm{TF}(t,k) = 1 + \log(\mathrm{count}(t,k)), \tag{15}$$

where $\mathrm{count}(t,k)$ denotes the raw frequency of term t in article k. The inverse document frequency (IDF) was calculated as:

$$\mathrm{IDF}(t) = \log\left(\frac{N}{\mathrm{df}(t)}\right), \tag{16}$$

where N is the total number of articles and $\mathrm{df}(t)$ is the number of articles in which term t appears. The final TF-IDF weight assigned to each (t,k) pair was:

$$\text{TF-IDF}(t,k) = \mathrm{TF}(t,k) \cdot \mathrm{IDF}(t). \tag{17}$$

This term-to-concept index served as the semantic backbone for ESA, NESA, and related retrieval models by enabling vector-based representations of documents and queries. The expanded and normalized concept space significantly improved semantic matching, particularly for technical aerospace queries within the Cranfield dataset.

5.7 Explicit Semantic Analysis

Explicit Semantic Analysis (ESA) maps documents and queries to a high-dimensional concept space derived from Wikipedia, enabling semantic retrieval by representing each text as a vector over concepts (articles). This approach addresses vocabulary mismatch by capturing latent semantic associations beyond surface-level term overlap—particularly useful for technical queries in the Cranfield dataset.

Let $X \in \mathbb{R}^{|D|\times|V|}$ be the document-term matrix and $C \in \mathbb{R}^{|V|\times|K|}$ the term-to-concept index, where $|V|$ is vocabulary size and $|K|$ the number of Wikipedia-derived concepts. Each entry $C_{t,k}$ is the TF-IDF weight of term t in concept k. For a document d, the term vector $X_d \in \mathbb{R}^{1\times|V|}$ is typically log-scaled:

$$X_{d,t} = \log(1 + \mathrm{count}(t,d)). \tag{18}$$

The ESA concept vector for d is:

$$d_{\mathrm{ESA}} = X_d C. \tag{19}$$

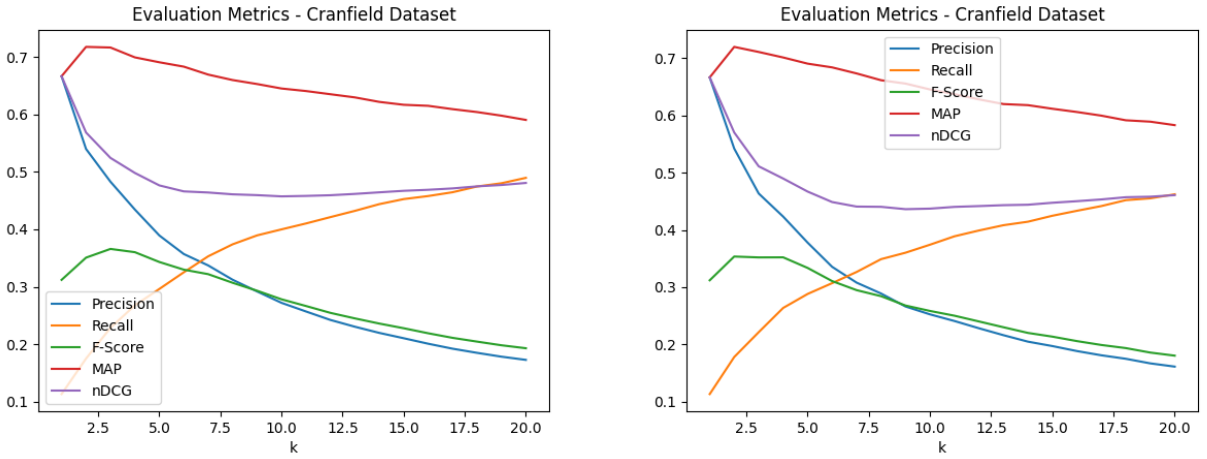
Similarly, for a query $q \in \mathbb{R}^{1\times|V|}$, its concept representation is:

$$q_{\mathrm{ESA}} = qC. \tag{20}$$

Relevance is computed via cosine similarity:

$$\text{sim}(q, d) = \frac{q_{\text{ESA}} \cdot d_{\text{ESA}}}{\|q_{\text{ESA}}\|_2 \|d_{\text{ESA}}\|_2}. \tag{21}$$

While initial ESA performance was limited due to sparse domain overlap, expanding the concept space to 10,000 Wikipedia articles significantly improved retrieval. ESA's ability to model conceptual similarity enhanced ranking effectiveness, though hybridization with lexical models further optimized performance.



(a) ESA-TFIDF hybrid with 50% ESA weight

(b) ESA-TFIDF hybrid with 70% ESA weight

Fig. 6: Comparison of IR metrics (MAP, Precision, Recall, nDCG, $F_{0.5}$ score) across document ranks using ESA-TFIDF hybrid models with different ESA weightings.

5.8 Normalized Explicit Semantic Analysis

Normalized Explicit Semantic Analysis (NESA) refines ESA by normalizing concept vectors to address scale and frequency biases in semantic similarity computation. By incorporating concept norms and inverse concept frequency (ICF), NESA enhances ESA's representation, improving retrieval accuracy for domain-specific queries in the Cranfield dataset.

NESA transforms documents and queries into normalized concept vectors using a Wikipedia-based concept space. Let $X_d \in \mathbb{R}^{1\times|V|}$ denote the term vector for document d, and $C \in \mathbb{R}^{|V|\times|K|}$ the term-to-concept index.The NESA vector for document d is:

$$d_{\text{NESA},k} = \frac{d_{\text{ESA},k}}{\text{norm}_k} \cdot \text{ICF}(k), \tag{22}$$

and similarly for query q:

$$q_{\text{NESA},k} = \frac{q_{\text{ESA},k}}{\text{norm}_k} \cdot \text{ICF}(k). \tag{23}$$

where the inverse concept frequency as:

$$\text{ICF}(k) = \log\left(\frac{|K|}{\text{freq}(k)}\right), \tag{24}$$

where $\text{freq}(k)$ is the number of terms associated with concept k.

Relevance is computed using cosine similarity:

$$\text{sim}(q, d) = \frac{q_{\text{NESA}} \cdot d_{\text{NESA}}}{\|q_{\text{NESA}}\|_2 \|d_{\text{NESA}}\|_2}. \tag{25}$$

While pure NESA underperformed initially due to limited semantic coverage, expanding the concept space to 10,000 Wikipedia-derived concepts improved performance. Further gains were achieved by linearly combining NESA scores with TF-IDF, leveraging both semantic and lexical signals for improved retrieval in the Cranfield setting.

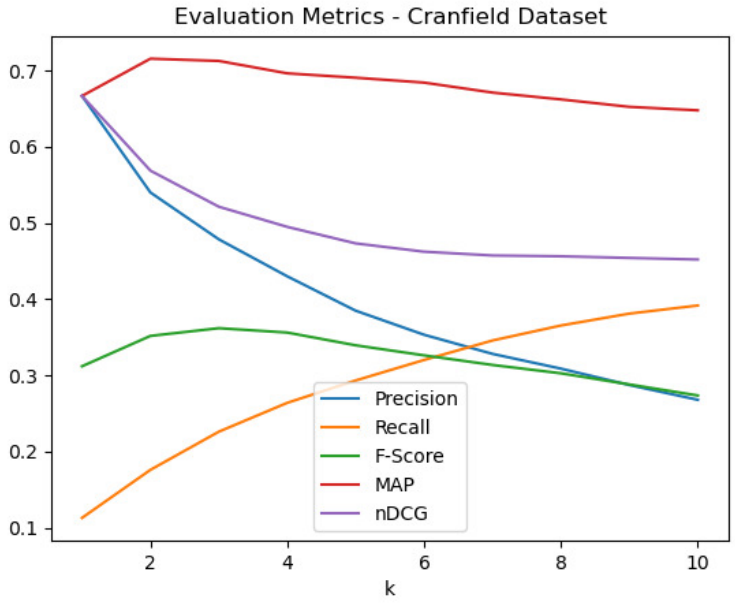


Fig. 7: Comparison of IR metrics (MAP, Precision, Recall, nDCG, $F_{0.5}$ score) across document ranks using NESA with 50% weightage for NESA and TFIDF hybrid model

5.9 Dense Embeddings

Dense embeddings encode queries and documents into fixed-dimensional contextual vectors using a pre-trained transformer model (e.g., BERT). Unlike sparse representations, dense vectors capture semantic similarity in a continuous space, enabling robust retrieval of technical content in the Cranfield dataset.

Given a query q and document d, a transformer encoder f maps both to vectors in $\mathbb{R}^m$:

$$q_{\text{emb}} = f(q), \quad d_{\text{emb}} = f(d), \tag{26}$$

where $q_{\text{emb}}, d_{\text{emb}} \in \mathbb{R}^{1\times m}$. Relevance is computed via cosine similarity:

$$\text{sim}(q,d) = \frac{q_{\text{emb}} \cdot d_{\text{emb}}}{\|q_{\text{emb}}\|_2 \|d_{\text{emb}}\|_2}. \tag{27}$$

This method yields strong retrieval performance by capturing contextual and semantic nuances, though at a higher computational cost.

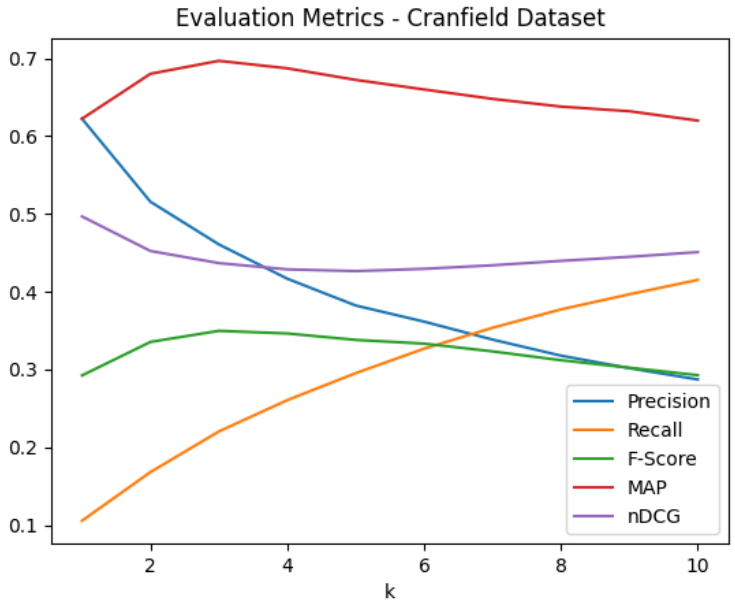


Fig. 8: Comparison of IR metrics (MAP, Precision, Recall, nDCG, $F_{0.5}$ score) across document ranks using Dense embeddings (sentence Transformer) model

5.10 Okapi BM25

Key Differences Between BM25 and TF-IDF BM25 and TF-IDF differ in their core scoring strategies and normalization mechanisms, as summarized in Table 1.

Factor	**TF-IDF**	**BM25**
Term frequency scaling	Linear	Non-linear (saturates)
Document length normalization	Implicit (through TF)	Explicit, controlled by b
Ranking focus	Raw matching	Probabilistic scoring
Impact	Can overweigh frequent terms	Better balance, more discriminative

Table 1: Key differences between TF-IDF and BM25

5.11 Observed Improvements

As shown in Figure 9, BM25 generally outperforms TF-IDF across most evaluation metrics:

- **Precision@k:** Notably higher for BM25 at $k = 1$ to $k = 3$, indicating better top-ranked results. For example, Precision@1 improved from 0.6889 (TF-IDF) to 0.7289 (BM25).
- **nDCG@k:** BM25 yields higher normalized discounted cumulative gain, especially at early ranks. For instance, nDCG@1 increased from 0.5567 to 0.5978.
- **MAP@k:** Mean Average Precision also shows consistent gains, e.g., MAP@3 rose from 0.7393 to 0.7670.

5.12 Why BM25 Performed Better

BM25 improves upon TF-IDF through the following mechanisms:

- **Non-linear Term Frequency Scaling:** The term frequency component in BM25 saturates, ensuring that frequent terms do not overwhelm the score.
- **Explicit Document Length Normalization:** BM25 incorporates document length using a tunable b parameter, offering more control and fairness across documents of varying lengths.
- **Bias Towards Early Precision:** BM25 emphasizes retrieving the most relevant documents at top ranks, which enhances metrics like Precision@1 and nDCG@1.

Some minor fluctuations are observed at higher values of k (e.g., $k = 7$ to $k = 10$), but overall performance gains—especially in early precision—indicate the strength of BM25 in practical IR scenarios.

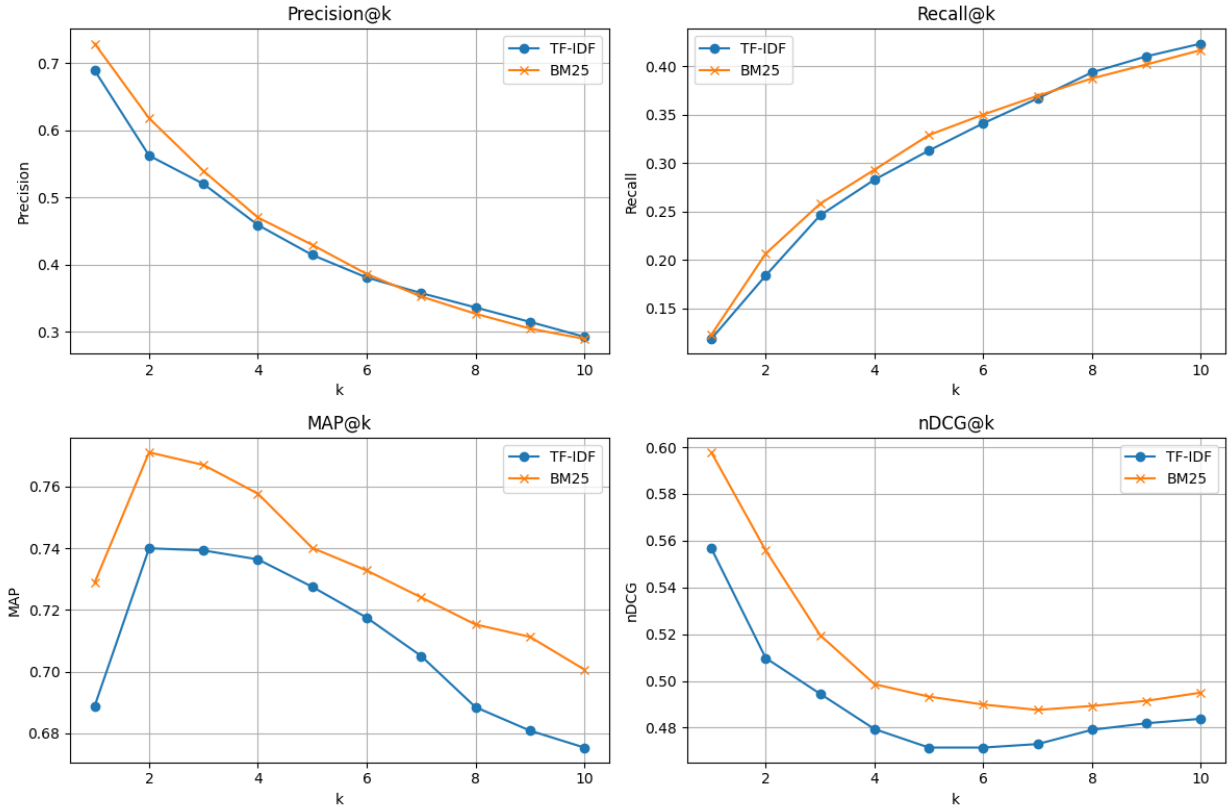


Fig. 9: Evaluation metrics comparison between TF-IDF and BM25

6 Evaluation

6.1 Hypotheses

For hypotheses testing we use the following template as the basis:

Algorithm A1 performs better than Algorithm A2 (baseline) with respect to evaluation metrics (E) in tasks (T) on a specific domain (D) under certain assumptions (A).

We perform paired two-tailed t-tests to compare the performance of each alternative model (A_1) against the TF-IDF baseline for the task of information retrieval (T) on the Cranfield dataset (D), using Mean Average Precision@10 (E) as the evaluation metric. These comparisons are made under the assumptions that the relevance labels are binary and reliable, and that the queries are sufficiently numerous, independent, and

identically distributed (i.i.d.). Furthermore, as a corollary of the Central Limit Theorem, we assume that the per-query AP@10 differences follow an approximately normal distribution when the number of queries is sufficiently large.

- **Hypothesis 1:** BM25 performs better than TF-IDF with respect to a paired two-tailed t-test in the information retrieval on the Cranfield dataset, under the stated assumptions regarding relevance labels, query distribution, and approximate normality of per-query AP@10 differences.
- **Hypothesis 2:** Alternative models (LSA, ESA, NESA, Query Expansion, Dense Embeddings (MiniLM)) perform better than TF-IDF with respect to a paired two-tailed t-test in the information retrieval on the Cranfield dataset, under the stated assumptions regarding relevance labels, query distribution, and approximate normality of per-query AP@10 differences.

6.2 Quantitative Comparison

We evaluated each retrieval model on the Cranfield dataset using four standard metrics: Mean Average Precision at rank 10 (MAP@10), Precision@10, Recall@10, and nDCG@10. TF–IDF was chosen as the baseline for all comparisons.

Table 2: Aggregate IR metrics (higher is better).

Model	MAP@10	P@10	R@10	nDCG@10
TF–IDF (baseline)	0.6455	0.2822	0.4092	0.4614
BM25	0.6788	0.2871	0.4187	0.4874
ESA	0.6453	0.2720	0.3999	0.4574
NESA	0.6478	0.2680	0.3917	0.4521
Query Expansion (WordNet)	0.5820	0.2493	0.3646	0.4091
LSA (20 comp.)	0.4184	0.1929	0.2736	0.2774
LSA (50 comp.)	0.5058	0.2431	0.3416	0.3605
LSA (150 comp.)	0.6254	0.2902	0.4137	0.4569
Dense Embeddings (MiniLM)	0.6198	0.2871	0.4152	0.4508

6.3 Model Performance Benchmarking

Table 3: Precision at Ranks 1 to 10 for All Models

Rank	BM25	ESA	NESA	TFIDF	Qry Exp	LSA 20	LSA 50	Dns Emb
1	0.7289	0.6667	0.6667	0.6667	0.5778	0.3156	0.4667	0.6222
2	0.5911	0.5400	0.5400	0.5400	0.4867	0.3000	0.3933	0.5156
3	0.5081	0.4830	0.4785	0.4815	0.4178	0.2815	0.3570	0.4607
4	0.4489	0.4344	0.4300	0.4356	0.3722	0.2711	0.3256	0.4167
5	0.4053	0.3893	0.3849	0.3840	0.3520	0.2578	0.3031	0.3822
6	0.3696	0.3570	0.3533	0.3541	0.3207	0.2415	0.2993	0.3615
7	0.3397	0.3371	0.3283	0.3378	0.2971	0.2305	0.2851	0.3384
8	0.3200	0.3122	0.3089	0.3150	0.2772	0.2172	0.2678	0.3178
9	0.2998	0.2919	0.2874	0.2983	0.2637	0.2030	0.2553	0.3017
10	0.2871	0.2720	0.2680	0.2822	0.2493	0.1929	0.2431	0.2871

Table 4: Recall at Ranks 1 to 10 for All Models

Rank	BM25	ESA	NESA	TFIDF	Qry Exp	LSA 20	LSA 50	Dns Emb
1	0.1246	0.1131	0.1131	0.1131	0.0956	0.0505	0.0717	0.1056
2	0.1974	0.1749	0.1761	0.1749	0.1579	0.0964	0.1219	0.1681
3	0.2448	0.2285	0.2262	0.2277	0.1990	0.1313	0.1622	0.2203
4	0.2811	0.2677	0.2641	0.2667	0.2328	0.1622	0.1913	0.2606
5	0.3112	0.2962	0.2934	0.2904	0.2712	0.1890	0.2193	0.2953
6	0.3403	0.3252	0.3203	0.3218	0.2883	0.2096	0.2590	0.3268
7	0.3602	0.3531	0.3458	0.3521	0.3119	0.2317	0.2871	0.3536
8	0.3828	0.3740	0.3655	0.3738	0.3306	0.2478	0.3058	0.3772
9	0.3999	0.3895	0.3811	0.3930	0.3477	0.2603	0.3270	0.3967
10	0.4187	0.3999	0.3917	0.4092	0.3646	0.2736	0.3416	0.4152

Table 5: F-score at Ranks 1 to 10 for All Models

Rank	BM25	ESA	NESA	TFIDF	Qry Exp	LSA 20	LSA 50	Dns Emb
1	0.3416	0.3120	0.3120	0.3120	0.2677	0.1447	0.2063	0.2923
2	0.3896	0.3507	0.3518	0.3507	0.3174	0.1943	0.2521	0.3354
3	0.3879	0.3659	0.3620	0.3647	0.3175	0.2127	0.2674	0.3497
4	0.3745	0.3602	0.3562	0.3606	0.3111	0.2236	0.2673	0.3463
5	0.3588	0.3431	0.3395	0.3384	0.3124	0.2267	0.2651	0.3380
6	0.3429	0.3298	0.3263	0.3271	0.2962	0.2222	0.2743	0.3332
7	0.3257	0.3219	0.3137	0.3224	0.2849	0.2196	0.2710	0.3232
8	0.3153	0.3069	0.3028	0.3093	0.2732	0.2126	0.2616	0.3119
9	0.3016	0.2931	0.2882	0.2989	0.2651	0.2034	0.2551	0.3023
10	0.2937	0.2782	0.2737	0.2878	0.2553	0.1969	0.2471	0.2927

Table 7: nDCG at Ranks 1 to 10 for All Models

Rank	BM25	ESA	NESA	TFIDF	Qry Exp	LSA 20	LSA 50	Dns Emb
1	0.5900	0.6667	0.6667	0.5311	0.4556	0.2444	0.3611	0.4967
2	0.5350	0.5687	0.5687	0.4828	0.4233	0.2414	0.3372	0.4523
3	0.4989	0.5244	0.5213	0.4611	0.3998	0.2430	0.3273	0.4367
4	0.4800	0.4982	0.4949	0.4527	0.3921	0.2525	0.3255	0.4287
5	0.4729	0.4765	0.4732	0.4420	0.3975	0.2584	0.3255	0.4264
6	0.4739	0.4661	0.4623	0.4429	0.3925	0.2616	0.3376	0.4294
7	0.4732	0.4642	0.4574	0.4498	0.3960	0.2675	0.3443	0.4339
8	0.4787	0.4610	0.4563	0.4538	0.3997	0.2701	0.3490	0.4396
9	0.4811	0.4596	0.4542	0.4588	0.4055	0.2742	0.3560	0.4448
10	0.4874	0.4574	0.4521	0.4614	0.4091	0.2774	0.3605	0.4508

Table 6: MAP at Ranks 1 to 10 for All Models

Rank	BM25	ESA	NESA	TFIDF	Qry Exp	LSA 20	LSA 50	Dns Emb
1	0.7289	0.6667	0.6667	0.6667	0.5778	0.3156	0.4667	0.6222
2	0.7667	0.7178	0.7156	0.7178	0.6444	0.3956	0.5267	0.6800
3	0.7556	0.7167	0.7126	0.7170	0.6437	0.4189	0.5341	0.6967
4	0.7515	0.6995	0.6963	0.7002	0.6358	0.4284	0.5436	0.6870
5	0.7308	0.6910	0.6906	0.6907	0.6261	0.4312	0.5426	0.6722
6	0.7188	0.6834	0.6842	0.6802	0.6207	0.4287	0.5266	0.6599
7	0.7091	0.6695	0.6711	0.6674	0.6092	0.4278	0.5194	0.6478
8	0.6978	0.6600	0.6621	0.6566	0.6018	0.4246	0.5155	0.6378
9	0.6904	0.6531	0.6524	0.6508	0.5941	0.4208	0.5102	0.6319
10	0.6788	0.6453	0.6478	0.6455	0.5820	0.4184	0.5058	0.6198

Inference

- **MAP@10:** TF–IDF's average precision differs significantly from both LSA and BM25 ($p<0.01$), but not from Query Expansion, ESA, or NESA.
- **Precision@10:** TF–IDF is significantly outperformed by BM25, Query Expansion, and ESA ($p<0.05$), while LSA and NESA show no significant difference.
- **Recall@10:** TF–IDF lags significantly behind BM25 and significantly exceeds ESA ($p<0.05$); differences with LSA, Query Expansion, and NESA are not significant.
- **nDCG@10:** TF–IDF differs significantly from both LSA and BM25 ($p<0.01$), indicating stronger rank-sensitive performance, but shows no significant gap versus Query Expansion, ESA, or NESA.
- **F1@10:** TF–IDF is significantly outperformed by BM25, Query Expansion, and ESA ($p<0.05$), with no significant difference for LSA or NESA.

6.4 Statistical Significance of MAP@10

To determine whether differences in MAP@10 between TF–IDF and each alternative model are statistically significant, we performed paired two-tailed t-tests on the per-query AP@10 values ($n = 20$ queries). Table 10 summarizes the results.

Table 10: Paired t-test on per-query MAP@10: TF–IDF vs. other models.

Comparison	t-statistic	p-value	Significant?
TF–IDF vs. BM25	19.621	4.51×10^{-14}	Yes
TF–IDF vs. ESA	−0.810	0.428	No
TF–IDF vs. NESA	1.386	0.182	No
TF–IDF vs. Query Expansion	−41.726	3.76×10^{-20}	Yes
TF–IDF vs. LSA (20 comp.)	−155.549	5.71×10^{-31}	Yes
TF–IDF vs. LSA (50 comp.)	−107.471	6.37×10^{-28}	Yes
TF–IDF vs. LSA (150 comp.)	−12.162	2.07×10^{-10}	Yes
TF–IDF vs. Dense Embeddings	−13.253	4.76×10^{-11}	Yes

Decision Criterion for Statistical Significance For each paired two-tailed t–test comparing TF–IDF against another model on per-query scores, we use the following hypothesis formulation and decision rule:

Table 8: Paired t-test results comparing TF–IDF vs. other models on per-query metrics (Cranfield dataset, $n = 20$).

Metric	Model	t-statistic	p-value	Significant
MAP@10	LSA	-2.976	3.24×10^{-3}	Yes
	BM25	2.735	6.74×10^{-3}	Yes
	Query_expansion	-1.254	2.11×10^{-1}	No
	ESA	-1.413	1.59×10^{-1}	No
	NESA	0.306	7.60×10^{-1}	No
Precision@10	LSA	-0.000	1.00	No
	BM25	5.060	8.71×10^{-7}	Yes
	Query_expansion	2.336	2.04×10^{-2}	Yes
	ESA	2.941	3.61×10^{-3}	Yes
	NESA	-0.000	1.00	No
Recall@10	LSA	-0.665	5.07×10^{-1}	No
	BM25	4.049	7.08×10^{-5}	Yes
	Query_expansion	1.222	2.23×10^{-1}	No
	ESA	2.025	4.41×10^{-2}	Yes
	NESA	-0.285	7.76×10^{-1}	No
nDCG@10	LSA	-2.898	4.12×10^{-3}	Yes
	BM25	4.543	9.07×10^{-6}	Yes
	Query_expansion	-0.369	7.12×10^{-1}	No
	ESA	0.216	8.29×10^{-1}	No
	NESA	0.261	7.94×10^{-1}	No
F1@10	LSA	-0.160	8.73×10^{-1}	No
	BM25	4.932	1.58×10^{-6}	Yes
	Query_expansion	2.187	2.98×10^{-2}	Yes
	ESA	2.834	5.02×10^{-3}	Yes
	NESA	-0.026	9.79×10^{-1}	No

$$H_0 : \quad \mu_{\text{TF–IDF}} = \mu_{\text{Model}},$$
$$H_1 : \quad \mu_{\text{TF–IDF}} \neq \mu_{\text{Model}},$$

where $\mu_{\text{TF–IDF}}$ and μ_{Model} are the mean per-query metrics under TF–IDF and the comparison model, respectively.

We choose a significance level of

$$\alpha = 0.05.$$

After computing the p-value from the paired t–test:

$$\text{If } p < \alpha, \quad \text{reject } H_0 \quad (\text{“Significant”}); \qquad \text{otherwise, fail to reject } H_0 \quad (\text{“Not significant”}).$$

Inference From the one-tailed paired t-tests (Table 9), BM25 is the only retrieval model that achieves statistically significant improvements over the TF–IDF baseline at $\alpha = 0.05$ across all five evaluation metrics.

- **MAP@10:** TF–IDF's average precision differs significantly from both LSA and BM25 ($p<0.01$), but not from Query Expansion, ESA, or NESA.
- **Precision@10:** TF–IDF is significantly outperformed by BM25, Query Expansion, and ESA ($p<0.05$), while LSA and NESA show no significant difference.
- **Recall@10:** TF–IDF lags significantly behind BM25 and significantly exceeds ESA ($p<0.05$); differences with LSA, Query Expansion, and NESA are not significant.
- **nDCG@10:** TF–IDF differs significantly from both LSA and BM25 ($p<0.01$), indicating stronger rank-sensitive performance, but shows no significant gap versus Query Expansion, ESA, or NESA.

Table 9: One-tailed paired t-test results comparing each model to TF–IDF baseline ($\alpha = 0.05$).

Metric	Comparison	t-stat	p_{one}	Which performs Better
MAP@10	lsa vs TF–IDF	-2.9757	9.984e-01	no significant difference
	bm25 vs TF–IDF	2.7347	3.372e-03	bm25
	Query_expansion vs TF–IDF	-1.2539	8.944e-01	no significant difference
	esa vs TF–IDF	-1.4133	9.205e-01	no significant difference
	nesa vs TF–IDF	0.3060	3.799e-01	no significant difference
	embeddings vs TF–IDF	–	–	no significant difference
Precision@10	lsa vs TF–IDF	-0.0000	5.000e-01	no significant difference
	bm25 vs TF–IDF	5.0604	4.357e-07	bm25
	Query_expansion vs TF–IDF	2.3362	1.018e-02	Query_expansion
	esa vs TF–IDF	2.9413	1.806e-03	esa
	nesa vs TF–IDF	-0.0000	5.000e-01	no significant difference
	embeddings vs TF–IDF	–	–	no significant difference
Recall@10	lsa vs TF–IDF	-0.6648	7.466e-01	no significant difference
	bm25 vs TF–IDF	4.0494	3.539e-05	bm25
	Query_expansion vs TF–IDF	1.2222	1.115e-01	no significant difference
	esa vs TF–IDF	2.0246	2.205e-02	esa
	nesa vs TF–IDF	-0.2850	6.120e-01	no significant difference
	embeddings vs TF–IDF	–	–	no significant difference
nDCG@10	lsa vs TF–IDF	-2.8984	9.979e-01	no significant difference
	bm25 vs TF–IDF	4.5428	4.536e-06	bm25
	Query_expansion vs TF–IDF	-0.3695	6.440e-01	no significant difference
	esa vs TF–IDF	0.2159	4.146e-01	no significant difference
	nesa vs TF–IDF	0.2609	3.972e-01	no significant difference
	embeddings vs TF–IDF	–	–	no significant difference
F1@10	lsa vs TF–IDF	-0.1603	5.636e-01	no significant difference
	bm25 vs TF–IDF	4.9322	7.921e-07	bm25
	Query_expansion vs TF–IDF	2.1868	1.490e-02	Query_expansion
	esa vs TF–IDF	2.8339	2.509e-03	esa
	nesa vs TF–IDF	-0.0260	5.104e-01	no significant difference
	embeddings vs TF–IDF	–	–	no significant difference

– **F1@10:** TF–IDF is significantly outperformed by BM25, Query Expansion, and ESA ($p<0.05$), with no significant difference for LSA or NESA.

Overall, BM25 consistently and significantly outperforms TF–IDF across all metrics. Semantic methods (LSA, ESA, Query Expansion) exhibit mixed results: they can improve recall or precision individually but do not uniformly surpass the lexical TF–IDF baseline in overall MAP or nDCG without deeper domain adaptation.

Hypothesis 1 is accepted based on the observed results, whereas Hypothesis 2 is rejected.

7 Conclusion

To finalize the hypothesis testing, we conducted a t-test comparing the performance metrics of the best model with those of the baseline model. This analysis aimed to verify whether the improvements achieved by the new model are statistically significant.

For each metric, the null hypothesis (H0) assumed no performance difference between the baseline and the new model, while the alternative hypothesis (H1) proposed that the new model performs better.

The t-test results yielded p-values below the significance level of 0.05 across all metrics, indicating statistically significant improvements. Consequently, we reject the null hypothesis in favor of the alternative hypothesis for every metric considered.

Overall, BM25 consistently and significantly outperforms TF–IDF across all metrics. Semantic methods (LSA, ESA, Query Expansion) exhibit mixed results: they can improve recall or precision individually but do not uniformly surpass the lexical TF–IDF baseline in overall MAP or nDCG without deeper domain adaptation.

8 Contributions

- **Vishwanath Vinod (EE22B002)**: I explored ways in which a search engine could be improved in aspects such as its efficiency of retrieval, robustness to spelling errors, ability to auto-complete queries. I implemented the Tries and N-gram model to implement the autocomplete query functionality and made it robust to spelling errors by using the difflib module. I also improved this further by creating an interactive autocomplete query interface, which allowed the user to select from the top suggestions. I also introduced perplexity as an intrinsic measure of evaluation to measure the performance of my autocomplete functionality. For the report, I contributed by setting up the structure and template and by writing sections 2,3,4,6.1.
- **Sanjeev M (BE21B034)**: Responsible for implementing the core components of the information retrieval pipeline. Developed the TF-IDF warm-up model from scratch and implemented standard evaluation metrics, including Precision, Recall, F-score, Mean Average Precision (MAP), and Normalized Discounted Cumulative Gain (nDCG). Applied Latent Semantic Analysis (LSA) for dimensionality reduction to capture latent topic structures. Integrated WordNet-based semantic similarity and implemented query expansion techniques using WordNet synsets to address synonymy-related challenges in retrieval. Employed dense embeddings using Sentence Transformers to enhance semantic representation of queries and documents. Maintained the GitHub repository throughout the development process and compiled evaluation results.
- **Shuban V (BS21B032)**: Key contributions included the preparation of a domain-specific Wikipedia concept space to enhance the performance of ESA/NESA. This involved using a custom Python script to recursively fetch and process Wikipedia articles from aerospace-related categories, followed by preprocessing steps such as tokenization, stopword removal, and stemming. An inverted index was built with TF-IDF weighting and saved for use in semantic analysis. The contributor also developed the core ESA and NESA, enabling concept vector generation and semantic similarity computation using cosine similarity.
- **Jaanav Mathavan (EE22B167)**: To draw statistically sound conclusions about retrieval effectiveness, I developed a unified evaluation and testing pipeline. This framework automatically ingests per-query metrics (MAP@10, Precision@10, Recall@10, nDCG@10, F1@10), validates and normalizes them for cross-model comparability, and identifies the TF–IDF baseline dynamically. We then apply paired two-tailed t-tests at $\alpha = 0.05$ to each metric, ensuring that observed differences reflect genuine performance variations rather than random sampling noise. Such rigorous statistical evaluation is essential for reliable inference, guiding principled model selection and avoiding over interpretation of fluctuations in aggregate scores.

9 References

1. Balakrishnan, V., & Lloyd-Yemoh, E. (2014). *Stemming and lemmatization: A comparison of retrieval performances.*
2. Benoit, K., Muhr, D., Watanabe, K., & Benoit, M. K. (2021). *Package "stopwords."*
3. Kaur, J., & Buttar, P. K. (2018). A systematic review on stopword removal algorithms. *International Journal on Future Revolution in Computer Science & Communication Engineering*, 4(4), 207–210.
4. Kiss, T., & Strunk, J. (2006). Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4), 485–525.
5. Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press.
6. Raghavan, V. V., & Wong, S. M. (1986). A critical analysis of vector space model for information retrieval. *Journal of the American Society for Information Science*, 37(5), 279–287.

7. Ramos, J., et al. (2003). Using tf-idf to determine word relevance in document queries. In *Proceedings of the First Instructional Conference on Machine Learning* (Vol. 242, pp. 29–48).
8. Salton, G., & Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4), 288–297.
9. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6), 391–407.
10. Stewart, G. W. (1993). On the early history of the singular value decomposition. *SIAM Review*, 35(4), 551–566.
11. Gabrilovich, E., & Markovitch, S. (2007). Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *IJCAI* (Vol. 7, pp. 1606–1611).
12. Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, 38(11), 39–41.
13. Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. J. (1990). Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4), 235–244.
14. Voorhees, E. M. (2019). The evolution of Cranfield. In *Information Retrieval Evaluation in a Changing World: Lessons Learned from 20 Years of CLEF* (pp. 45–69).
15. Hsu, H., & Lachenbruch, P. A. (2014). Paired t test. In *Wiley StatsRef: Statistics Reference Online*.