IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 28, Issue 3, Ser. II (May. – June. 2026), PP 10-13
www.iosrjournals.org

Abstract-Based Research Article Filtering Using Term Frequency-Inverse Document Frequency and Cosine Similarity

Mohammad Jahangir, Edip Senyurek
Student, Department of Computer Engineering, Vistula University, Warsaw, Poland
Head of Department, Department of Computer Engineering, Vistula University, Warsaw, Poland.

Abstract
Abstracts are transformed into Term Frequency Inverse Document Frequency (TF-IDF) vectors and relevance is estimated by cosine similarity against a user query or a target abstract. The method is designed for simple deployment, transparent scoring, and practical use in small to medium collections. Experiments on 5,000 abstracts from arXiv data which is available in Huggingfaces an open online repository of scientific papers, show that TF-IDF with cosine similarity outperforms a keyword overlap baseline. Precision at the top five results improved from 0.265 to 0.327, meaning the proposed method retrieved more relevant papers near the top of the ranked list. The mean reciprocal rank improved from 0.479 to 0.548, meaning the first relevant result appeared earlier in the ranking.

Keywords: *Term Frequency-Inverse Document Frequency, Information Retrieval, Abstract Similarity, Cosine Similarity, Content-Based Filtering, Document Ranking*

Date of Submission: 25-05-2026 Date of Acceptance: 03-06-2026

I. INTRODUCTION

Digital libraries and scholarly search engines provide access to large collections, but researchers still spend substantial time sifting through results that match a keyword yet miss the actual research intent. Titles can be unclear and author keywords are often incomplete. In contrast, abstracts are concise summaries that better represent the contribution and context of the paper. This motivates abstract-level filtering as a practical first stage for literature review and topic exploration.

Classic information retrieval methods model a document as a bag of words and score relevance using term-weighting schemes such as Term Frequency-Inverse Document Frequency (TF-IDF). Term Frequency measures how often a word appears in one document. Inverse Document Frequency measures how rare or important that word is across the full collection of documents. TF-IDF is attractive because it is computationally efficient, requires no supervised training data, and provides transparent explanations: a paper scores higher because it shares informative terms with the query and those terms are rare across the corpus.

This work focuses on a simple pipeline that (1) preprocesses abstracts, (2) computes TF-IDF vectors, (3) ranks documents by cosine similarity to a query or a target abstract, and (4) evaluates the ranking quality. The goal is not to replace semantic embedding models, but to provide a strong baseline and a practical solution for constrained settings where interpretability and low complexity matter.

Contributions

1. A clear end-to-end workflow for abstract-based filtering using Term Frequency-Inverse Document Frequency and cosine similarity.
2. A discussion of normalization/standardization choices that affect ranking stability.
3. An evaluation template (metrics and labeling protocol) that can be applied to any local corpus.

II. RELATED WORK

Term weighting and vector space retrieval have a long history in information retrieval. Early work formalised the intuition behind Inverse Document Frequency as a measure of term specificity. TF-IDF and related term-weighting variants continue to be used as baselines in document ranking, clustering, and classification. Although modern neural approaches can capture semantics beyond exact term overlap, TF-IDF remains competitive in many domains, especially with limited data, specialised terminology, or when explainability is required.

Best Matching 25 (BM25), a probabilistic search ranking method, is widely used in production search

systems and often outperforms plain TF-IDF on longer documents. However, for short texts such as abstracts and for small corpora, TF-IDF combined with cosine similarity is frequently sufficient and easier to implement.

III. METHODOLOGY

3.1 Dataset and Problem Definition

Let the corpus contain *N* research articles, where *N* means the total number of articles. Each article is represented by its abstract. In formula form, the full document collection is written as:

D = {d1, d2, ..., dn}.

Here, **D** means the whole dataset, and **d1, d2, ..., dn** mean individual abstracts inside that dataset. We consider two use cases:

1. **Query-to-abstract retrieval:** a user supplies a query *q*, where *q* means a topic description or a set of keywords.
2. **Target-abstract retrieval:** a user supplies a target abstract *t*, where *t* means one selected abstract used to find similar abstracts.

The output is a ranked list of documents ordered by a relevance score. A higher relevance score means that the abstract is more similar to the query or target abstract.

3.2 Text Preprocessing

To reduce noise while preserving meaning, the following preprocessing steps are recommended:

1. Lowercasing
2. Removing punctuation and extra whitespace
3. Stop-word removal, which means removing very common words such as "the", "is", and "and"
4. Optional lemmatization or stemming, which means reducing related word forms to a common form
5. Optional bigrams, which means two-word phrases such as "neural network" and "information retrieval"

3.3 Term Frequency-Inverse Document Frequency Vector Construction

For each document *d* and term *w*:

- Term Frequency:

tf(w, d) = count(w in d)

- Inverse Document Frequency (smoothed):

idf(w, D) = log((N + 1) / (df(w) + 1)) + 1

where **df(w)** means document frequency, or the number of documents that contain the word *w*. The Term Frequency-Inverse Document Frequency weight is:

tfidf(w, d) = tf(w, d) × idf(w, D)

In simple words, a word receives a high TF-IDF value when it appears often in one abstract but does not appear in many other abstracts. Each abstract is represented as a vector, which is a numerical list of TF-IDF values for its words.

3.4 Similarity Estimation (Cosine Similarity)

To rank documents against a query or target abstract, we compute cosine similarity:

sim(a, b) = (a · b) / (||a|| × ||b||)

where *a* and *b* are TF-IDF vectors for two abstracts or for one query and one abstract.

Cosine similarity is preferred because it reduces the effect of document length: two abstracts can be similar even if one is longer, as long as the direction (term distribution) is similar.

3.5 Score Normalization Options

In practice, abstract lengths and vocabulary sizes vary, which can affect raw similarity scores. Two common strategies are:

1. **Vector length normalization:** ensures each abstract vector is scaled to the same length before similarity is computed. This prevents longer abstracts from receiving higher scores only because they contain more words.
2. **Standardization of similarity scores:** converts scores into a common scale, for example by using a z-score. A z-score shows whether a value is above or below the average score.

This paper recommends vector length normalization as a default. Standardization is optional and should be used only when the application needs a fixed relevance threshold, such as keeping only abstracts above a chosen score.

3.6 Algorithm (Ranking Pipeline)

Input: Corpus abstracts **D** (the full abstract collection), query **q** (the user's search topic), or target abstract **t** (the selected abstract used for comparison)
Output: Ranked list of documents **R**, where **R** means the final ordered result list

1. Preprocess all abstracts in **D**
2. Build the TF-IDF vocabulary from **D**
3. Transform **D** into a TF-IDF matrix **X**, where **X** is the table of numerical word weights for all abstracts
4. Transform **q** (or **t**) into a TF-IDF query vector **xq**, where **xq** is the numerical representation of the search topic
5. Compute cosine similarity between **xq** and each row of **X**
6. Sort documents by similarity score (descending)

IV. EXPERIMENTAL SETUP

4.1 Corpus

We evaluated the proposed method on a public corpus of scientific abstracts sourced from arXiv metadata. arXiv is an open online repository where researchers share scientific papers. The experiments used a subset of 5,000 abstracts from the sondalex/arxiv-abstracts-2021-embeddings-10000 dataset. This dataset is derived from gfissore/arxiv-abstracts-2021 and is released under Creative Commons Zero (CC0), meaning it can be reused freely. Each record includes an abstract field called content and one or more arXiv category tags.

4.2 Query Selection and Relevance Labels

We considered the **target-abstract retrieval** scenario: each query is one abstract from the corpus, and the goal is to retrieve other abstracts on the same topic. Since manual relevance labeling is expensive, we used arXiv category tags as weak relevance labels.

To avoid trivial cases where a very broad category dominates the corpus, we evaluated only on queries whose category frequency in the corpus fell within a bounded range. Specifically, a query abstract was eligible if it had at least one category whose corpus frequency was between 30 and 150 documents (inclusive). For each eligible query, documents sharing any of its eligible categories were treated as **relevant**.

In total, 300 eligible queries were sampled with a fixed random seed. A fixed random seed means the same random sample can be repeated again. The mean size of the relevant set per query was 103.31 documents, meaning that each query had about 103 relevant abstracts on average. The median was 127 documents, the minimum was 48 documents, and the maximum was 258 documents.

4.3 Metrics

We report standard ranking metrics averaged across queries:

- Precision at 5 and Precision at 10: the percentage of relevant abstracts in the first 5 or first 10 search results.
- Mean Reciprocal Rank: a score that shows how early the first relevant result appears. A higher value is better.
- Normalised Discounted Cumulative Gain at 10: a score that checks whether relevant results are placed near the top of the first 10 results. A higher value is better.

4.4 Compared Methods

We compared three methods:

1. **Keyword overlap:** binary bag-of-words overlap score, meaning the method counts whether the same words appear in both texts.
2. **TF-IDF using dot product:** TF-IDF vectors without vector length normalization, scored by dot product. Dot product is a direct numerical comparison between two vectors.
3. **TF-IDF with cosine similarity (proposed):** vector-normalized TF-IDF vectors scored by cosine similarity. This is the proposed method because it reduces the effect of abstract length.

All methods used English stop-word removal. TF-IDF used unigrams and bigrams. A unigram is one word, and a bigram is a two-word phrase. The setting min_df=2 means a word or phrase must appear in at least two documents to be included. The setting max_df=0.9 means a word or phrase is removed if it appears in more than 90 percent of the documents. Sublinear term frequency scaling was used to reduce the impact of repeated words.

V. RESULTS AND DISCUSSION

5.1 Quantitative Results

Table 1 summarizes the retrieval performance. TF-IDF consistently outperformed keyword overlap. Cosine similarity further improved ranking quality compared to raw TF-IDF dot product, indicating that length-invariant scoring provides a more stable notion of similarity for abstracts.

Table 1: Retrieval performance (mean across 300 queries)

Method	Precision at 5	Precision at 10	Mean Reciprocal Rank	Normalized Discounted Cumulative Gain at 10
Keyword overlap	0.265	0.211	0.479	0.241
TF-IDF using dot product	0.309	0.268	0.524	0.292
TF-IDF with cosine similarity	0.327	0.282	0.548	0.307

The values in the table are between 0 and 1. A higher value means better retrieval performance. For example, a Precision at 5 value of 0.327 means that, on average, 32.7 percent of the first five returned abstracts were relevant. This is approximately

1.64 relevant abstracts in the first five results. A Precision at 10 value of 0.282 means that about 28.2 percent of the first ten results were relevant. Mean Reciprocal Rank of 0.548 means that the first relevant result usually appeared close to the top of the ranked list. Normalized Discounted Cumulative Gain at 10 of 0.307 means that the proposed method placed more relevant abstracts near the top of the first ten results than the baseline methods.

5.2 Discussion

The results suggest three practical observations.

1. **Term weighting matters:** TF-IDF reduces the impact of common words and highlights terms that better characterize the abstract, improving over keyword overlap.
2. **Normalization improves robustness:** Cosine similarity with vector length normalization slightly but consistently improves Precision at K and Mean Reciprocal Rank relative to raw TF-IDF dot product. Here, K means the number of top results being checked, such as 5 or 10.
3. **Weak labels are useful but imperfect:** Category tags provide an efficient proxy for relevance, yet two abstracts can share a category and still be conceptually distant, while related papers may use different category tags. Manual evaluation on a smaller set of queries remains a recommended follow-up for deployment.

VI. CONCLUSION

This paper presents an interpretable, lightweight method for filtering and ranking research articles based on abstract similarity. By representing abstracts and queries as Term Frequency-Inverse Document Frequency vectors and scoring them with cosine similarity, the method provides a practical baseline for literature screening in small-to-medium corpora. The workflow is easy to reproduce and can be adapted through domain stop-words, n-grams, and parameter tuning. Future work includes integrating synonym expansion, comparing against Best Matching 25 and neural embedding models, and evaluating at a larger scale with more diverse queries and multi-annotator relevance labels.

REFERENCES

[1]. K. Spärck Jones, "A Statistical Interpretation of Term Specificity and Its Application in Retrieval,".
[2]. G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval,"
[3]. C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*.
[4]. S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford, "Okapi at TREC-3," in *Proceedings of the Third Text REtrieval Conference (TREC-3)*, NIST, 1995.
[5]. Scikit-learn: Machine Learning in Python.
[6]. https://scispace.com/pdf/identifying-the-development-and-application-of-artificial-4pfg5m58k5.pdf
[7]. https://huggingface.co/datasets/common-pile/arxiv_abstracts_filtered