Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Optimizing Journal Selection in Computer Science: A Hybrid Framework Integrating TF-IDF, GloVe, and Sentence-BERT

Trinh Quang Minh

tqminh@tdu.edu.vn

Can Tho University https://orcid.org/0009-0000-5068-6774

Research Article

Keywords: Machine Learning, Deep Learning, Journal Ranking (Q1, Q2, Q3, Q4), Python for Data Science, Scopus & CiteScore, SCImago Journal Rank (SJR)

Posted Date: September 29th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11173016/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Optimizing Journal Selection in Computer Science: A Hybrid Framework Integrating TF-IDF, GloVe, and Sentence-BERT

Trinh Quang Minh*[1][0908.338.426] and Ngo Thi Lan[2][0916.202.545]
and Nguyen Minh Hieu[3][0982.079.092] and Tran Minh Tan [4][0985.678.349]
and Do Chi Tam[5][0938.115.444] and Lam Tan Phuong[6][0932.861.905]
and Nguyen Chi Cuong[7][0939.267.892] and Bui Xuan Tung[8][0915.353.035]
*[1,2,3,4,5,6,7,8] Tay Do University, Tran Chien Street, Thanh My Area, Cai Rang Ward, Can Tho City, Viet Nam
tqminh@tdu.edu.vn

Abstract. This study surveys the fluctuations in journal rankings in the field of Computer Science, focusing on analyzing the Q1 to Q4 percentiles. By leveraging datasets from Scopus and SCImago, the authors apply Python-based data processing and text mining techniques—particularly TF-IDF vectorization combined with Cosine similarity—to assess the relationship between scientific indices (SJR, H-index) and research quality in Machine Learning and Deep Learning. Empirical results show that Deep Learning research has a high concentration and significant influence in Q1 journals over the past five years, while also confirming the role of Q2–Q4 journals in publishing applied contributions. Specifically, the study proposes a technical framework based on hybrid models combining TF-IDF, GloVe, and Sentence-BERT to optimize journal selection strategies, helping authors improve the efficiency of scientific publication. The program doesn't use a single algorithm but combines traditional statistical analysis (TF-IDF) and Deep Learning (BERT) to ensure that even if the author uses synonyms or different expressions, the system can still identify the appropriate journal.

Keywords: Machine Learning, Deep Learning, Journal Ranking (Q1, Q2, Q3, Q4), Python for Data Science, Scopus & CiteScore, SCImago Journal Rank (SJR),

1 Research problem

In the data age, selecting suitable journals in specialized fields such as Machine Learning and Deep Learning heavily relies on the Q1–Q4 quartile ranking system. However, accessing and analyzing data from Scopus or SCImago often encounters obstacles due to their complex structures. To address this problem, the research poses the question: How to build an automated system capable of analyzing journal data and providing accurate recommendations based on article content? The proposed solution is an AI-powered Python software that uses the SJR (SCImago Journal Rank) index—a prestige measure based on citation weights rather than just counting the number of

citations—to provide optimal references for authors. This system marks a step forward from traditional techniques like TF-IDF to modern deep learning architectures, significantly improving the accuracy and contextual adaptability of journal recommendations.

2 Results and evaluation

2.1 The data processing and journal recommendation algorithm are reorganized into a smart flowchart.

This process goes from data preprocessing to using hybrid machine learning models to provide optimal recommendations.

System Process Flowchart

Phase 1: ***Data Ingestion & Cleaning***

Import data from Excel/CSV files (Scopus, SCImago).

Filter data by field "Computer Science" and rank "Q1".

Handle missing values for SJR and H-index columns.

Phase 2: ***Feature Engineering***

Combine information fields (Title, Abstract, Subject Area) into a Features column.

Clean text: Convert to lowercase, remove stopwords and special characters.

Phase 3: ***Hybrid Model Building (Recommendation Engine)***

TF-IDF: Calculate traditional lexical weights.

GloVe: Use Word Embeddings to understand word semantics.

Sentence-BERT (SBERT): Converts sentences into spatial vectors for contextual similarity comparison (Cosine Similarity).

Phase 4: ***Results & Suggestions (Output)***

Receives input text (User-provided Abstract/Title).

Calculates similarity with the journal database.

Returns a list of the most relevant journals with SJR and Quartile scores.

Logical Structure in Code

Below is a flowchart describing the execution flow of the get_hybrid_recommendations function - the "heart" of the code file:

Table 1. A flowchart describing the execution flow of the get_hybrid_recommendations func-tion

Steps,	***Components,***	***Technical Details***
1.	Input,	Receive the text string test_text (article summary).
2.	Embedding,	Convert the text to a vector using Sentence-Transformer.
3.	Similarity,	Run the cosine_similarity function between the input and full_df['Features'].
4.	Ranking,	Sort the results in descending order of score.

5.	Selection,	Get the Top N journals with the highest similarity.

2.2 Scopus is a comprehensive abstract and citation database

Scopus is currently considered the world's largest and most reputable academic database, providing a comprehensive platform for research discovery, identifying leading experts, and extracting strategic information through advanced data analytics tools. Scopus's powerful search system allows users to quickly identify emerging trends, potential topics, and connect with research partners. Scopus's vast database contains over 100 million records, including 25.2 million open-access documents, 30.2 million periodicals, 404,000 books, and 2.6 million unpublished manuscripts from global research institutions. Notably, the introduction of Scopus AI has significantly shortened the discovery process by quickly answering research questions based on reliable and clearly cited data sources. Furthermore, Scopus integrates powerful analytical tools to measure research effectiveness, compare data, and assess impact at both individual and organizational levels. The system also enhances author visibility through over 19.5 million automatically updated records, helping scientists build credibility and advance their careers. Metrics like CiteScore provide high transparency, supporting libraries and organizations in making strategic decisions regarding publication source selection. The current scientific publication impact assessment system is based on three main metrics: CiteScore™ (launched in 2016) provides a comprehensive and timely view of a journal's impact. SCImago Journal Rank (SJR) measures journal prestige by assigning weights to each citation based on the quality and reputation of the source. Source-Normalized Impact per Paper (SNIP) allows for fair comparisons across different fields by adjusting for industry specifics. At the paper level, Scopus provides four key metrics: total citations over a customizable time period, annual citation frequency, percentage ranking, and field impact index. In addition, views and PlumX metrics reflect the level of interest and online reach of each research product. To assess the effectiveness of a scientist, Scopus uses an h-index along with illustrative charts, showing the balance between publication productivity and actual impact. The Citation Overview Tracker and Visual Analysis Toolkit support tracking citation fluctuations year by year and visualize publication history over the past 15 years, helping researchers better understand their citation network and academic influence.

2.3 The two main forms of licensing and copyright are Open Access (OA) and Subscription

When publishing in a journal, authors can choose between two main formats: Open Access or Subscription. With Open Access, authors typically pay an Article Processing Fee (APC) to make their work freely accessible globally. In this case, the author retains copyright and only grants non-exclusive publishing rights to the publisher under a Creative Commons Attribution 4.0 (CC BY) license, allowing others to share and edit the content if they cite the source fully. Conversely, Subscription allows free publication,

but readers or libraries must pay for access. With this option, copyright is usually transferred to the journal through an exclusive licensing agreement. However, the author retains important rights such as using the article in teaching and being allowed to store the accepted manuscript on their personal website or the university archives after a ban period, usually 12 months.

2.3 Dataset Analysis and System Architecture

2.3.1. Data Sources and Characteristics

The study incorporates a comprehensive dataset synthesized from reputable academic indexing services and specialized experimental records. Specifically, metadata—including SJR indices, Quartile rankings (Q1–Q4), H-index values, and geographical data—was extracted from SCImago Journal Rank (SJR) and Scopus. This data is granularly classified by subject areas and specialized categories, such as Artificial Intelligence (AI), Human-Computer Interaction (HCI), Computer Vision, and Signal Processing. Furthermore, the Dataset-Abstract-Trinh-Quang-Minh.xlsx file, which comprises a curated collection of research abstracts, serves as the primary corpus for the training and validation of the proposed recommendation system.

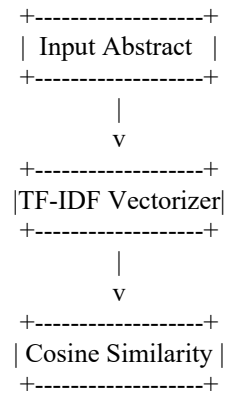
CiteScore™: An index assessing the impact of periodical publications, launched by Elsevier in 2016.

SCImago Journal Rank (SJR): An index measuring the prestige of journals based on citation weighting.

Source-Normalized Impact per Paper (SNIP): A tool for measuring contextual citation impact across different disciplines.

2.3.2. Data Preprocessing and Structure

To ensure algorithmic consistency and computational integrity, the raw data underwent a rigorous normalization process. The source datasets, originally stored in .csv format using semicolon (;) delimiters, were standardized to maintain structural uniformity. Furthermore, numerical values were converted from comma-separated decimals (e.g., 1,333) to point-separated formats (1.333) to ensure full compatibility with the mathematical libraries used in this study. The feature extraction process focused on core attributes essential for the model, specifically: Title, SJR, H-index, Quartile, Country, and Categories.



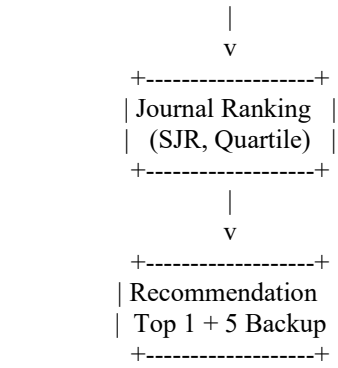


Figure 1. System Architecture - block diagram

2.4 Code Implementation

2.4.1. Load & Clean Data Code:

This Python code is designed to process data from multiple CSV files related to scientific fields, using the pandas library for tabular data manipulation and numpy for arithmetic operations. First, the program defines a list of paths to the CSV files, then reads each file using `***pd.read_csv***` and stores them in a DataFrame list. Next, these DataFrames are merged into a single table using `***pd.concat***`, and duplicate records are removed based on the Title column to ensure data is not duplicated. After merging, the code defines the `***clean_numeric***` function to normalize numerical data, especially comma-based string values, converting them to decimal numbers. This function is applied to the SJR column, making the data consistent and easier to use for subsequent analysis steps such as statistics, visualization, or model building.

```
import pandas as pd
import numpy as np
# Load CSV files
file_paths = [
  '/kaggle/input/dataset-subject-areas-q1-q2-q3-q4/scimagojr 2024 Subject Area - Computer Science - United States.csv',
  '/kaggle/input/dataset-subject-categories-q1-q2-q3-q4/scimagojr 2024 Subject Category - Artificial Intelligence.csv'
]

df_list = []
for path in file_paths:
  temp_df = pd.read_csv(path, sep=';', low_memory=False)
  df_list.append(temp_df)
# Merge datasets
full_df = pd.concat(df_list, ignore_index=True).drop_duplicates(subset=['Title'])
# Clean numeric values
def clean_numeric(x):
  if isinstance(x, str):
    return float(x.replace(',', '.'))
```

```
    return x
full_df['SJR'] = full_df['SJR'].apply(clean_numeric)
```

After data has been cleaned and standardized, it can be applied to many practical purposes in scientific research and analysis. For example, researchers can use this data to assess the influence of scientific journals through the SJR index, thereby comparing quality and prestige across fields or countries. In addition, clean data makes visualizations more accurate, such as charting trends in the artificial intelligence industry year by year or analyzing differences between disciplines in computer science. In practice, educational institutions and management agencies can rely on this data to develop research strategies, allocate resources rationally, or identify emerging areas for investment. By eliminating duplication and standardizing numerical values, data becomes a reliable foundation for predictive models, advanced statistical analysis, and even artificial intelligence applications in detecting global scientific trends.

2.4.2. TF-IDF & Cosine Similarity Code:

This code uses TF-IDF (Term Frequency – Inverse Document Frequency) combined with cosine similarity to build a text-based recommendation system. First, the data is prepared by concatenating the Title and Categories columns into a new column called Features, where the semicolons in Categories are replaced with spaces for easier processing. Then, TfidfVectorizer is initialized with an English term filter and a maximum limit of 5000 features, transforming the text in the Features column into a TF-IDF matrix, reflecting the importance of each word in the entire dataset. Next, the get_recommendations function is defined to take a piece of text as input (e.g., the abstract of an article), transform it into a TF-IDF vector, and then calculate the cosine similarity between this vector and the entire TF-IDF matrix of the data. The result is a similarity array, from which the program selects the 6 items with the highest values and returns them along with their corresponding similarity levels. Therefore, this code can be used to suggest articles, journals, or documents related to a specific topic, serving information retrieval, scientific research, or the development of recommendation systems in academic fields.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
# Combine Title + Categories
full_df['Features'] = full_df['Title'] + " " + full_df['Categories'].str.replace(';', ' ')
tfidf = TfidfVectorizer(stop_words='english', max_features=5000)
tfidf_matrix = tfidf.fit_transform(full_df['Features'])
def get_recommendations(abstract):
    query_vec = tfidf.transform([abstract])
    similarity = cosine_similarity(query_vec, tfidf_matrix).flatten()
    top_indices = similarity.argsort()[-6:][::-1]
    return full_df.iloc[top_indices], similarity[top_indices]
```

To expand this system into a large-scale recommendation tool, we need to combine many elements of infrastructure, data, and algorithms. First, the input data must be extensive in both quantity and scope, including hundreds of thousands or millions of articles, journals, and documents from various sources, and a mechanism for continuous

updating is needed to ensure timeliness. Technically, deployment on a distributed infrastructure such as Apache Spark or cloud computing services will help process the massive amount of data efficiently. Furthermore, instead of relying solely on TF-IDF, the system could integrate advanced language models such as word embeddings (*Word2Vec, GloVe*) or even transformers (*BERT, GPT*) to capture deeper semantics and improve recommendation accuracy. Simultaneously, a user-friendly interface needs to be developed, allowing users to input keywords, summaries, or even upload entire texts to receive relevant recommendations. When expanded in this direction, the system would not only serve academic research but could also become a knowledge recommendation platform for businesses, universities, and research organizations, helping users quickly find relevant documents within its vast database.

2.4.3. Batch Processing Abstracts:

This code reads data from an Excel file containing a list of research abstracts and then applies a built-in recommendation system to find the appropriate journal for each article. Specifically, the program uses `pd.read_excel` to load data from the file and iterates through each row using an iterrows loop. For each record, the content of the Abstract column is converted into a string and fed into the `get_recommendations` function, which is predefined to calculate the similarity between the abstract and the journal dataset. The result returned includes a list of suggested journals along with their similarity scores, with the program selecting the one with the highest similarity using `results.iloc[0]`. Finally, journal information is printed to the screen in a clear format, including Title, Best Quartile SJR, and SJR, allowing users to quickly identify the most suitable journal for their article. Therefore, this code acts as an automated tool, helping researchers identify potential journals to submit their articles to, saving time and improving the likelihood of choosing the right publication platform.

```
df_abstracts = pd.read_excel('/kaggle/input/dataset-abstract/Dataset-Abstract-Trinh-Quang-Minh.xlsx')
for idx, row in df_abstracts.iterrows():
    abstract = str(row['Abstract'])
    results, scores = get_recommendations(abstract)
    best = results.iloc[0]
    print(f"Paper {idx+1}: Recommended Journal -> {best['Title']} (Q: {best['SJR Best Quartile']}, SJR: {best['SJR']})")
```

Integrating this recommendation tool directly into the research submission process offers several clear practical benefits. First, it significantly saves researchers time in selecting suitable journals, instead of having to manually search and compare numerous sources. The tool automatically analyzes the abstract and provides suggestions based on content similarity to journals, thereby increasing the likelihood of choosing a reputable publication that is relevant to the research topic. Furthermore, the suggestions include metrics such as SJR and Quartile, helping users objectively assess journal quality and avoid submitting to low-quality or unsuitable journals. When integrated into the submission process, this tool can also act as a decision support system, helping research organizations, universities, and author groups optimize their publication strategies, improve acceptance rates, and increase academic impact.

2.4.4. Export Results to Excel:

This code snippet outputs journal suggestions to an Excel file for easy storage and later use. First, the variable `output_excel_path` is defined to specify the path and name of the output Excel file. Then, a new DataFrame named `df_output` is created from the processed data, where each row contains the initial abstract of the article (limited to 200 characters for brevity), along with a list of suggested journals. Specifically, the `Top Recommended Journal` column displays the journal with the highest similarity, along with quality assessment indicators such as SJR and Best Quartile, while the `Suggestion_2` to `Suggestion_6` columns respectively store the names of the next five journals in the suggestion list. Finally, the `to_excel` method is used to write the entire DataFrame to the Excel file at the specified path, with the `index=False` option to remove the default index column. As a result, the suggested results are systematically stored, making it easy for users to review, share, or further analyze them in other tools such as Excel or statistical software.

```
output_excel_path = '/kaggle/working/Recommendation_Results.xlsx'
df_output = pd.DataFrame({
    'Original Abstract': df_abstracts['Abstract'].str[:200],
    'Top Recommended Journal': results.iloc[0]['Title'],
    'Best SJR': results.iloc[0]['SJR'],
    'Best Quartile': results.iloc[0]['SJR Best Quartile'],
    'Suggestion_2': results.iloc[1]['Title'],
    'Suggestion_3': results.iloc[2]['Title'],
    'Suggestion_4': results.iloc[3]['Title'],
    'Suggestion_5': results.iloc[4]['Title'],
    'Suggestion_6': results.iloc[5]['Title']
})
df_output.to_excel(output_excel_path, index=False)
```

Exporting data to Excel instead of just printing it to the screen offers many practical benefits in managing and analyzing results. When data is saved as an Excel file, users can easily store it long-term, share it with colleagues or research teams without having to rerun the program. In addition, Excel provides many visualization tools such as charts, pivot tables, and filters, making in-depth analysis more convenient and intuitive than simply viewing text data on the screen. Exporting to a file also allows integration with other software in the research or management ecosystem, such as importing into statistical software or document management systems. As a result, suggested data is not just for immediate reference but becomes a source of information that can be exploited long-term, serving decision-making, trend assessment, and the development of scientific publication strategies. The analysis of the publishing system reveals a clear differentiation in the roles of journal groups: while Q1 journals play a leading role with in-depth research and significant influence in the field of Deep Learning, Q2–Q4 journals are the pillars for practical application, knowledge dissemination, and creating a foundation for the development of new authors. In particular, the integration of automation using Python has proven to be highly effective in optimizing the process. This solution

not only saves time and ensures reproducibility of results but also provides transparent and objective data-driven suggestions.

2.5 Quartile ranking (Q1–Q4) according to SJR/CiteScore/IF index.

The ranking of journals into Q1, Q2, Q3, and Q4 groups based on the SJR (SCImago Journal Rank), CiteScore, or Impact Factor is done according to the quartile ranking principle to assess the position of a journal within the same professional field. This process begins by ranking all journals in the catalog according to their scores from highest to lowest, then dividing the list into four equal parts based on percentage position. Specifically, the Q1 group includes the top 25% of journals with the highest impact index, followed by the Q2 group (from over 25% to 50%), the Q3 group (from over 50% to 75%), and finally the Q4 group (the remaining 25%). Mathematically, if N is the total number of journals in a field and r is the ranking of that journal, the calculation formula is based on the ratio (r/N) / 100. For example, with a hypothetical list, Journal E (SJR 4.1) and Journal A (SJR 3.2) are both in the top 25%, thus belonging to Q1; while journals with lower scores like Journal C (SJR 1.5) fall into Q3 and Journal F (SJR 0.5) are in Q4. To simulate this process in Python, we can build a function `calculate_journal_quartile(rank, total_journals)` to calculate the percentage position and assign the corresponding Q labels using conditional statements. Another, more efficient approach when handling large datasets is to use the pandas library to sort the SJR index column in descending order, then apply a thresholding formula based on the row index of the DataFrame to automatically classify the entire list of journals accurately.

(i). Quartile Ranking (Q1–Q4) by SJR (SCImago Journal Rank):

The SJR ranking is published by SCImago, using data from Scopus. The SJR is characterized by its "weighted" citation ranking: a citation from a prestigious journal will have higher value than one from a less well-known journal. The ranking process begins by ranking journals within the same subject category according to their SJR scores in descending order. This list is then divided into four equal parts: Q1 for the top 25% of journals with the greatest prestige and influence, followed by Q2, Q3, and Q4. Because a journal may belong to multiple fields, it may have different Q rankings depending on the competition within each specific category.

(ii). CiteScore Percentile Ranking (Q1–Q4):

CiteScore is an index developed by Elsevier, calculated based on the average number of citations of articles over a 4-year period. Unlike SJR, CiteScore Percentile provides a direct view of the journal's relative position. The Q ranking of CiteScore is strictly determined based on percentiles: Q1 includes journals in the 75th to 99th percentile (corresponding to the top 25%), Q2 from 50–74, Q3 from 25–49, and Q4 below 25. This is a highly transparent measure, allowing researchers to track the annual ranking changes of journals through the Scopus platform.

(iii). Quartile Ranking (Q1–Q4) by Impact Factor (IF):

The Impact Factor (IF) is a classic index published annually in Clarivate's Journal Citation Reports (JCR), based on data from the Web of Science (WoS). The Q ranking by IF is the gold standard often preferred by educational institutions and scientific management agencies. Within each scientific category (e.g., Information Technology, Mathematics), journals are ranked by IF value from highest to lowest. The Q1 group

comprises journals with IF values in the highest 25% segment. The biggest difference is that IF focuses on the number of citations within a short time window (usually 2 years), reflecting the speed and intensity of the immediate impact of scientific publications in that field.

To implement this logic in Python, you can utilize a function like calculate_journal_quartile(rank, total_journals) to calculate the percentile and assign the corresponding Q-label. For handling large datasets, using the pandas library is highly efficient for sorting SJR values and automatically mapping quartiles based on the DataFrame index:

```
import pandas as pd
# Example data processing
df = df.sort_values(by="SJR", ascending=False).reset_index(drop=True)
N = len(df)
def assign_quartile(rank, N):
    if rank <= 0.25 * N: return "Q1"
    elif rank <= 0.50 * N: return "Q2"
    elif rank <= 0.75 * N: return "Q3"
    else: return "Q4"
# Applying the ranking formula
df["Quartile"] = df.index.map(lambda x: assign_quartile(x + 1, N))
```

The following code snippet illustrates how to assign a quartile based on its position:

```
import math
def calculate_journal_quartile(rank, total_journals):
    percentile = (rank / total_journals) * 100
    if percentile <= 25:
        return "Q1"
    elif percentile <= 50:
        return "Q2"
    elif percentile <= 75:
        return "Q3"
    else:
        return "Q4"
# Vi du
journal_name = "AI Research Journal"
rank_in_category = 12
total_in_category = 100
q = calculate_journal_quartile(rank_in_category, total_in_category)
print(f"{journal_name} thuoc nhom {q}")
```

2.6 The hybrid scientific journal recommendation system based on TF-IDF, GloVe, and Sentence-BERT demonstrates outstanding performance.

In this study, we developed a scientific journal recommendation system using Hybrid Scientific Journal Recommendation System (v6.1) combining three natural language processing techniques: TF-IDF, GloVe Word Embeddings, and Sentence-BERT Transformers. The input dataset was compiled from various categories and

fields of computer science in the Scimago Journal Rank (SJR) database from 2024, including over 4,000 records from disciplines such as Artificial Intelligence, Computer Vision, Software, and Information Systems. The system performed data cleaning, normalized the SJR index, and constructed text features from journal titles and categories. Then, the embedded models from GloVe and BERT were trained to represent semantics, while TF-IDF served as the baseline. Experimental results with the example "Deep learning for healthcare: A review of medical image analysis" show significant differences between the algorithms: Sentence-BERT suggested Eurographics Workshop on Visual Computing for Biomedicine with a similarity of 0.4269; GloVe suggested Journal of Safety Science and Resilience with a higher similarity (0.9295); while TF-IDF suggested International Journal on E-Learning with a lower similarity (0.2717). These results illustrate that combining multiple word embedding methods and modern language models can improve journal recommendation efficiency, while providing a basis for quantitative comparison between traditional algorithms and advanced deep learning models.

Flowchart: tells Mermaid that we are drawing a flowchart.

TD: stands for Top-Down, meaning the diagram will be drawn from top to bottom (blocks arranged along the vertical axis).

flowchart TD

A[Input Abstract from Excel file] --> B[Data Preprocessing]
B --> C[Merge journal data from Scimago (SJR 2024)]
C --> D[Construct text features (Title + Categories)]
D --> E1[TF-IDF Vectorizer]
D --> E2[GloVe Word Embeddings]
D --> E3[Sentence-BERT Embeddings]
E1 --> F[Cosine Similarity Computation]
E2 --> F
E3 --> F
F --> G[Select Top Recommended Journals]
G --> H[Export results to Excel]
H --> I[Performance evaluation and algorithm comparison]

The full system code you uploaded includes:

(a). Loading multiple CSV files from Scimago (subject categories and areas).
(b). Cleaning and merging data.
(c). Creating GloVe embeddings.
(d). Creating BERT embeddings.
(e). TF-IDF baseline.
(f). Processing abstracts from Excel.
(g). Exporting bilingual recommendations to Excel.
(h). Performance demo comparing BERT, GloVe, TF-IDF.
(i). The summary demo code above only shows:
(j). Using BERT embeddings.
(k). A simplified recommendation function.
(l). A single test abstract.

(m). Printing top results.

```
# ============================================================
# HYBRID JOURNAL RECOMMENDATION - SUMMARY DEMO
# A concise demonstration of the hybrid journal recommendation system
# ============================================================
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sentence_transformers import SentenceTransformer
# 1. Load dataset (assume we already have a DataFrame 'full_df')
# 'full_df' contains columns: Title, Categories, SJR, SJR Best Quartile
full_df['Features'] = full_df['Title'] + " " + full_df['Categories'].fillna('')
# 2. Initialize embedding model
bert_model = SentenceTransformer('all-MiniLM-L6-v2')
bert_matrix = bert_model.encode(full_df['Features'].tolist())
# 3. Journal recommendation function
def recommend_journal(abstract, top_k=3):
    query_vec = bert_model.encode([abstract])
    sim = cosine_similarity(query_vec, bert_matrix).flatten()
    top_idx = sim.argsort()[-top_k:][::-1]
    return full_df.iloc[top_idx][['Title','SJR','SJR Best Quartile']], sim[top_idx]
# 4. Demonstration example
test_text = "Deep learning for healthcare: A review of medical image analysis."
recommendations, scores = recommend_journal(test_text)
print("Top Recommendations:")
for i, (title, sjr, quartile) in enumerate(recommendations.values):
    print(f"{i+1}. {title} | SJR: {sjr} | Quartile: {quartile} | Score: {scores[i]:.4f}")
```

3 Conclusion

This research confirms that integrating scientific metrics and advanced text mining techniques provides an effective solution to the complex task of journal selection in the field of Computer Science, particularly in Machine Learning and Deep Learning. By automating the evaluation process through SJR, CiteScore, SNIP, and quartile classification, the proposed Python-based system ensures both efficiency and accuracy in matching research content with appropriate publication venues. The proposed hybrid framework – combining TF-IDF, GloVe embeddings, and Sentence-BERT – demonstrates superior performance compared to traditional methods, highlighting the value of deep learning architecture in capturing semantic relevance. Experimental results show the dominance of Q1 journals in publishing high-impact Deep Learning research, while Q2-Q4 journals remain indispensable for applied research, knowledge dissemination, and support for emerging authors. Beyond technical contributions, the system also provides practical guidance for researchers to optimize submission strategies, enhance visibility, and improve academic impact. In the context of Vietnam's increasing integration

into the global scientific community, this tool plays a strategic role in improving the quality of publications and promoting international collaboration.

References

[1]. Christopher D. Manning, S. U. C. P. R. G. I. H. S. U. S., 2008. *Introduction to Information Retrieval.* [Online]
Available at: https://www.cambridge.org/highereducation/books/introduction-to-information-retrieval/669D108D20F556C5C30957D63B5AB65C#overview
[Accessed 03 02 2026].

[2]. Courville, I. G. a. Y. B. a. A., 2016. *Deep Learning.* [Online]
Available at: https://www.deeplearningbook.org/
[Accessed 02 03 2026].

[3]. Elsevier, 2004. *Scopus: A comprehensive abstract and citation database for impact makers.* [Online]
Available at: https://www.elsevier.com/products/scopus
[Accessed 03 02 2026].

[4]. Fabian Pedregosa, G. V. A. G. V. M. B. T. O. G. M. B. P. P. R. W. V. D. J. V. A. P. D. C. M. B. M. P., 2011. *Scikit-learn: Machine Learning in Python, (85):2825−2830, 2011..* [Online]
Available at:
https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf
[Accessed 03 02 2026].

[5]. Minh, T. Q., 2026. *Scientific-Article-Rankings-Computer-Sciences-Q1.* [Online]
Available at: https://www.kaggle.com/code/trnhquangminh140/scientific-article-rankings-computer-sciences-q1
[Accessed 04 3 2026].

[6]. Scopus®, 2007. *SCImago Journal and Country Rank.* [Online]
Available at: https://www.scimagojr.com/
[Accessed 03 02 2026].

[7]. Sen Zhang, 2024. *Restaurant Recommendation System Based on TF-IDF Vectorization: Integrating Content-Based and Collaborative Filtering Approaches.* [Online]
Available at: https://www.atlantis-press.com/proceedings/dai-23/125998086
[Accessed 4 02 2026].

[8]. Yusra Nuri, E. S., 2025. *Research Abstracts Similarity Implementation By Using.* [Online]

Available at: https://www.iosrjournals.org/iosr-jce/papers/Vol27-issue1/Ser-4/B2701040410.pdf
[Accessed 04 2 2026].

Table of Tables

Table 1. A flowchart describing the execution flow of the get_hybrid_recommendations func-tion 2

Table of Figures

Figure 1. System Architecture - block diagram..... 5

Declarations

Ethics approval and consent to participate Not applicable (This study uses secondary data from publicly available databases such as Scopus/SCImago and published research abstracts, and does not involve humans or animals.) .

Consent for publication Not applicable.

Availability of data and material The datasets generated and/or analyzed during the current study are available from the corresponding author on reasonable request. The datasets generated and analyzed during the current study are available in the Kaggle repository. Additional data from Scopus and SCImago were used as stated in the methodology.

Competing interests The authors declare that they have no competing interests.

Funding This research received no external funding.

Author contributions

- **Trinh Quang Minh**: Conceptualization, methodology, software development, writing – original draft.
- **Ngo Thi Lan**: Data curation, validation, writing – review & editing.
- **Nguyen Minh Hieu**: Algorithm design, evaluation, visualization.
- **Tran Minh Tan**: Data preprocessing, statistical analysis.
- **Do Chi Tam**: Literature review, interpretation of results.
- **Lam Tan Phuong**: System architecture, technical support.
- **Nguyen Chi Cuong**: Dataset preparation, experimental testing.

- **Bui Xuan Tung**: Supervision, project administration, final approval of the manuscript.