Evaluating Recommender Systems for Digital Library Datasets

Ákos Lévárdy*

Faculty of Informatics and Information Technologies STU in Bratislava

Abstract. As digital content continues to grow rapidly, users face increasing difficulty in finding relevant information efficiently—especially within large-scale digital libraries. Recommender systems help address this challenge by delivering personalized suggestions that improve user experience and reduce information overload. This study focuses on evaluating content-based recommender systems tailored for digital library datasets. These systems analyze textual data from books to generate recommendations based on similarities in content features. The goal is to compare algorithms such as TF-IDF, LSA, GloVe, FastText and Sentence-BERT under different settings, assessing their effectiveness in generating Top-N recommendations. To provide a comprehensive evaluation, the study uses multiple performance metrics including similarity, diversity, confidence, and coverage, while also measuring execution time and memory usage. We benchmark each model using metrics such as similarity, confidence, diversity, and coverage, along with computational performance indicators like runtime and memory usage. Our findings show that while transformer-based and neural embeddings (e.g., FastText, BERT) achieve higher similarity scores, traditional methods like LSA strike a better balance between similarity, diversity, and efficiency. These results provide actionable insights into algorithm selection for scalable recommendation systems in academic libraries.

Keywords: Recommender System · Content-Based · Evaluation · Performance Metrics · Digital Library.

1 Introduction

Making decisions in today's digital world is not always easy. Whether choosing a product, a movie, or a travel destination, people are often faced with an overwhelming number of options and varying levels of information and trustworthiness. While some users know exactly what they are looking for and seek immediate answers, others are open to exploring new possibilities and expanding their knowledge [4].

* Bachelor study programme in field: Informatics
Supervisor: PaedDr. Pavol Baťalík, Faculty of Informatics and Information Technologies STU in Bratislava

Recommendation Systems (RS) are designed to ease this process by predicting useful items, comparing them, and suggesting the most similar options based on user preferences. These systems have become essential tools for reducing information overload, especially with the rapid growth of big data [9]. By analyzing large volumes of textual data, they aim to understand users' preferences and generate personalized recommendations [16].

The task of providing users with available options that match their needs and interests is increasingly important in today's consumer society. Without a starting point—such as a list of relevant suggestions—users may feel overwhelmed or even choose to give up entirely. For example, trying to pick a movie from scratch without any recommendations can lead to decision fatigue. Recommender systems help prevent this by offering tailored suggestions that guide the user and reduce decision friction.

The core goal of RS is to personalize content and improve user experience by recommending books, movies, music, products, and more. For example, in a digital library which has loads of books available online, a recommender system might suggest books or even specific paragraphs based on a user's reading history [17]. These systems function by identifying relationships between users and the items they interact with—such as preferring historical documentaries over action films [2].

Various techniques are used to achieve this personalization. Collaborative filtering recommends items based on similar users' preferences, while content-based filtering analyzes item features like genre or keywords. Hybrid approaches combine both for more accurate results [3]. Regardless of the method, the aim is to help users discover relevant content they might not have found on their own.

That said, information systems must also account for the fact that user preferences can change over time. This phenomenon, known as concept drift, refers to unexpected changes in data patterns or behaviors that can significantly affect the system's prediction accuracy [15]. Recommender systems need to be adaptive in order to stay effective over long periods.

Recommendation systems differ from search engines, although both are used to navigate large amounts of information. While search engines retrieve content based on keywords, recommender systems make predictions based on user behavior and inferred interests [7].

This paper focuses on evaluating and comparing different recommendation algorithms for suggesting books and their parts based on textual content. Algorithms such as TF-IDF, LSA, and BERT will be tested to assess how effectively they generate recommendations. Evaluation will consider metrics like similarity, diversity, confidence, and coverage [8], along with performance factors such as execution time or memory usage.

Recommending items can be done in a variety of ways. Several types of recommendation systems exist, and their methods of operation differ [14]. Here are the different recommendation system types:

Content-Based Filtering recommends items based on a user's previous choices

or interactions by finding similarities between the items the user has shown interest in and other items with similar features (e.g., genre, attributes, keywords) [1]. This approach focuses primarily on the item's characteristics rather than relying on other users' preferences.

Collaborative Filtering relies more on preferences of other users and their behaviour. The point is that users who had similar interests before will have them again in the future for new items. This technique relies on having user related data, feedback that could be either explicit (ratings) or implicit (passive user behaviour) [13].

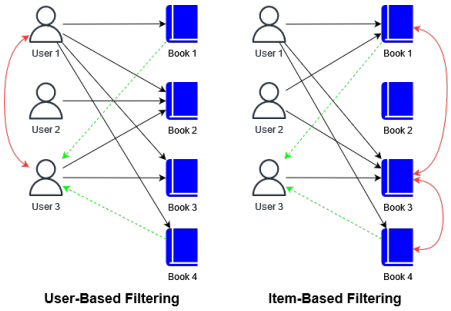

Fig. 1. Memory-based CF recommendation.

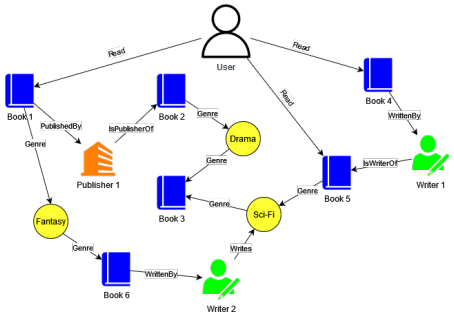

Fig. 2. KG-aware recommendation.

Knowledge-Graphs use a network of data where items are linked through their features. Showing how items relate to one another and connecting them with more information [10].

Context-Aware recommendation systems are adding contextual factors to the rating process, where the recommended item is based on the users explicit ratings, the items implicitly inferred ratings and also the contextual variables [9]. The variables for example when recommending a movie can be the location from where the user watches the movie, the time and the companion who the user watches the movie with.

Demographic recommendation systems are recommending items based on a demographic profile of the user. They categorize the users from their personal attributes and try to make user stereotypes [6].

Utility-Based systems generate the recommendations by computing the utility of each item for the user. The utility of an item refers to how valuable it is to a user and is calculated using a utility function which combines different factors of the user's preferences [6].

Deep Learning-Based are trying to find complex patterns in the users behaviour and the items features using deep learning algorithms and neural networks. These models can locate hidden links and can offer highly customized recommendations.

Hybrid methods try to combine the useful characteristics of both collabora-

tive filtering and content-based filtering methods. They take into account both the users past preferences and the preferences of other people who might share the users taste [12].
The most popular techniques used are the Collaborative Filtering, Content-Based Filtering and the Hybrid method [5]. This paper will focus on comparing different algortihms that are used for Content-Based Filtering, since trying to recommend books and their paragraphs in digital libraries relies on the textual content of them.

2 System Decription

Extracting Information and Building a Dataset

First, the books were extracted from the digital library and stored as individual JSON files. Each file contained the entire data of a single book, further segmented into sentences, pages, and paragraphs for flexibility in experimentation. For the full-text analysis, a dataset of 45k paragraphs was created. The average paragraph length was approximately 60-70 words, with the longest paragraph consisting of 2000 words.
In addition to the extracted book paragraphs, a publicly available dataset titled *Books Details Dataset*, which was scraped from Goodreads [11] was also used in the experiments. This dataset includes metadata for over 13,000 books, such as title, author, genres, ratings, and textual descriptions averaging approximately 163 words. These summaries were treated similarly to paragraphs during preprocessing and recommendation, allowing models to be tested on both short- and long-form content for robustness and comparison.

Models selected for the Evaluation

In the context of academic book recommendations, a variety of content-based algorithms can be used to extract meaningful patterns and relationships from textual data. These algorithms differ in their approach to representing and analyzing documents, words and textual data. The selected algorithms represent a diverse spectrum of CB recommendation approaches, from classical sparse models to modern transformer-based embeddings:

- **TF-IDF and BoW** are *sparse count-based models*, included as traditional baselines for textual similarity. They are fast, interpretable, and serve as a reference point for evaluating more advanced models.
- **LSA** is a *matrix factorization technique* that captures latent semantic structures in documents. It adds a dimension-reduction perspective to the evaluation.
- **GloVe and FastText** are *pretrained static word embedding models*, chosen for their ability to encode semantic meaning into dense vectors. FastText also includes subword information, allowing it to handle out-of-vocabulary words effectively.

– **Sentence-BERT (all-MiniLM-L6-v2)** is a *transformer-based deep learning model* specifically optimized for sentence-level semantic similarity. It represents a state-of-the-art approach in content-based recommendation and is included to benchmark modern deep embedding techniques.

This selection allows a fair comparison across traditional, embedding-based, and deep learning methods in terms of similarity, diversity, and computational efficiency.
To generate recommendations, each algorithm first transformed (embedded or vectorized) all paragraphs in the dataset into numerical representations according to its own architecture:

– **TF-IDF** and **BoW** used scikit-learn's `TfidfVectorizer` and `CountVectorizer` with English stop words removed. Tokenization was based on whitespace and punctuation, with a vocabulary size capped at 5,000 most frequent terms.
– **LSA** began with TF-IDF vectorization and then applied `TruncatedSVD` for dimensionality reduction with 500 components, uncovering latent semantic structures across terms.
– **GloVe** used pre-trained 50-dimensional word embeddings from the `glove.6B.50d.txt` file, which contains word vectors trained on 6 billion tokens from Wikipedia. Tokenization was done using simple whitespace splitting, and paragraph embeddings were obtained by averaging the vectors of all matching words in the vocabulary.
– **FastText** was trained on the dataset using `gensim`'s implementation with the following hyperparameters: `vector_size=100`, `window=5`, `min_count=2`, `workers=6`, and `epochs=20`. Paragraph vectors were computed by averaging all available subword embeddings from the tokenized input.
– **Sentence-BERT** used the pre-trained `all-MiniLM-L6-v2` model from the `SentenceTransformers` library. The model handled tokenization and embedding internally, producing 384-dimensional sentence embeddings via mean pooling. Embeddings were computed on GPU to accelerate cosine similarity calculations.

After generating embeddings or vectors for all documents, the input paragraph was transformed using the same method. Cosine similarity was calculated between the input and all candidate paragraphs to identify the most similar items. This metric measures the angle between two vectors, with a smaller angle (closer to 1) indicating greater similarity. To ensure efficiency, especially on larger datasets, the cosine similarity computations were parallelized using `joblib`. The sorted similarity scores were then used to return the most similar paragraphs as recommendations.
To control the quality of recommendations, only results with cosine similarity above a dynamically computed threshold were retained. This threshold was defined as a fraction (typically 50%) of the maximum similarity score observed in a given query. The rationale behind this relative threshold was to adapt the selection to each input case and filter out weak matches while preserving the top results. Finally, the remaining candidates were sorted by similarity score, and the

top-N most similar items were returned as recommendations. Notably, the similarity threshold varied across models. For TF-IDF, BoW, LSA, and BERT, the default threshold of 0.5 times the maximum similarity score was used. However, FastText and GloVe produced consistently high similarity scores across most inputs, often exceeding 0.9 even for weak semantic matches. This was likely due to their dense embeddings and averaging strategies, which compress semantic differences. Therefore, stricter thresholds were applied—0.95 for FastText and 0.99 for GloVe—to avoid returning overly generic or weakly relevant results. This threshold tuning was essential to ensure the coverage metric reflected meaningful recommendations and to improve result precision in models where cosine similarity values were inflated by design.

Table 1. Metrics Comparison by Algorithm – Book Descriptions

Metric	**TF-IDF**	**BoW**	**FastText**	**GloVe**	**LSA**	**BERT**
Avg Similarity	0.196	0.254	0.945	0.981	0.425	0.359
Coverage (%)	0.34	0.23	0.53	0.25	0.12	0.20
Confidence	0.95279	0.93772	0.99023	0.99787	0.93325	0.94778
Diversity	0.00270	0.00499	0.00010	0.00000	0.00551	0.00334
Elapsed Time (s)	3.973	15.256	49.959	14.888	13.248	74.456
Memory Usage (MB)	293.49	378.45	2002.80	420.78	744.64	1227.44
CPU Time (s)	1.729	0.809	44.964	10.314	8.179	103.399

Table 2. Metrics Comparison by Algorithm – Paragraphs

Metric	**TF-IDF**	**BoW**	**FastText**	**GloVe**	**LSA**	**BERT**
Avg Similarity	0.276	0.193	0.939	0.989	0.297	0.349
Coverage (%)	0.06	1.30	15.32	21.52	1.20	9.29
Confidence	0.95279	0.97252	0.99032	0.99799	0.95164	0.95391
Diversity	0.00270	0.00074	0.00010	0.00000	0.00273	0.00246
Elapsed Time (s)	15.206	3.722	13.373	4.646	3.501	21.217
Memory Usage (MB)	293.49	202.89	1193.43	344.33	424.85	1141.38
CPU Time (s)	7.230	0.275	11.821	3.210	2.041	27.757

3 Experimental Results for Evaluation

Each algorithm was evaluated using 10 independent test runs, where each run used a different input item. For paragraph-level evaluation, 10 distinct paragraphs from the extracted book texts were used as inputs. For the description-level evaluation, 10 different book summaries from the Goodreads dataset [11]

were selected. The same set of 10 inputs was used across all algorithms to ensure consistency and fairness.

In each run, the algorithm generated recommendations from the corresponding dataset (paragraphs or descriptions), and the similarity scores of these recommendations were used to compute performance metrics. The final metric values for each algorithm were obtained by averaging the scores across all 10 runs.

The following metrics were calculated:

- **Average Similarity**: Mean cosine similarity between the input item and its Top-N recommendations.
- **Item Coverage**: Percentage of unique recommended items across all runs, indicating how broadly the model explores the dataset.
- **Confidence**: Inverse of the standard deviation of similarity scores. A lower variance indicates more confident (i.e., stable) recommendations.
- **Diversity**: Variance of similarity scores among the recommended items. Higher variance implies greater diversity within the Top-N list.

These metrics were chosen to reflect both the quality and consistency of the recommendations, especially since explicit user feedback data was not available. Therefore, traditional recommendation metrics like Precision or Recall could not be applied. Instead, content-based internal evaluation using cosine similarity allowed for a fair, interpretable comparison of the models.

4 Future Work

Future research could expand the evaluation by including additional state-of-the-art models beyond the ones tested in this study. Transformer-based architectures such as RoBERTa, DistilBERT, or retrieval-augmented models like RAG and ColBERT may offer improved performance, especially on complex or ambiguous queries. These models are designed to capture deeper contextual relationships and may enhance both relevance and semantic precision in recommendations.

Another promising direction is to experiment with hybrid approaches that combine content-based filtering with collaborative signals (e.g., user ratings or interaction history), enabling systems to adapt dynamically to individual user preferences. This could lead to more personalized and context-aware recommendations.

From a technical perspective, optimizing runtime and memory efficiency, particularly for large-scale datasets, could be achieved through model distillation, approximate nearest neighbor techniques, or embedding quantization. Additionally, evaluating models across more diverse content types and multilingual datasets would help assess their generalizability and robustness.

5 Conclusions

This study presented a comparative evaluation of several content-based recommendation algorithms—namely TF-IDF, BoW, LSA, GloVe, FastText, and

Sentence-BERT—applied to digital library data at both paragraph and book-description levels. The results demonstrate that while neural models like FastText and GloVe achieve high similarity scores, they also require significantly more computational resources. On the other hand, traditional models such as TF-IDF and BoW offer fast and lightweight alternatives, but with reduced accuracy.

LSA was shown to strike a middle ground between relevance, diversity, and efficiency, making it a viable option for systems where balance across these factors is important. Sentence-BERT provided strong performance with moderate similarity and higher processing overhead, suggesting its potential for systems where context-rich recommendations are critical.

Overall, the analysis highlights clear trade-offs between recommendation quality, computational cost, and item coverage. These insights offer practical guidance for selecting and deploying content-based recommendation algorithms in academic or large-scale library systems. By aligning system requirements—such as speed, accuracy, or breadth of exploration—with algorithmic strengths, developers can build more effective and scalable recommender systems tailored to digital content.

References

1. Content-based Recommender Systems: State of the Art and Trends, pp. 73–105 (2010). https://doi.org/10.1007/978-0-387-85820-3_3, doi:10.1007/978-0-387-85820-3_3
2. Aggarwal, C.C.: An Introduction to Recommender Systems (2016). https://doi.org/10.1007/978-3-319-29659-3, doi:10.1007/978-3-319-29659-3
3. Aymen, A.T.M., Imène, S.: Scientific paper recommender systems: A review **361 LNNS**, 896 – 906 (2022). https://doi.org/10.1007/978-3-030-92038-8_92, doi:10.1007/978-3-030-92038-8_92
4. Blanco, R., Cambazoglu, B.B., Mika, P., Torzec, N.: Entity recommendations in web search **8219 LNCS**(PART 2), 33 – 48 (2013). https://doi.org/10.1007/978-3-642-41338-4_3, doi:10.1007/978-3-642-41338-4_3
5. B.Thorat, P., Goudar, R.M., Barve, S.: Survey on collaborative filtering, content-based filtering and hybrid recommendation system. International Journal of Computer Applications **110**(4), 31–36 (2015). https://doi.org/10.5120/19308-0760, doi:10.5120/19308-0760
6. Burke, R.: Hybrid recommender systems: Survey and experiments **12**(4), 331 – 370 (2002). https://doi.org/10.1023/A:1021240730564, doi:10.1023/A:1021240730564
7. De Nart, D., Tasso, C.: A personalized concept-driven recommender system for scientific libraries. vol. 38, p. 84 – 91 (2014). https://doi.org/10.1016/j.procs.2014.10.015, doi:10.1016/j.procs.2014.10.015
8. Gunawardana, A., Shani, G., Yogev, S.: Evaluating Recommender Systems (2022). https://doi.org/10.1007/978-1-0716-2197-4_15, doi:10.1007/978-1-0716-2197-4_15
9. Haruna, K., Ismail, M.A., Suhendroyono, S., Damiasih, D., Pierewan, A.C., Chiroma, H., Herawan, T.: Context-aware recommender system: A review of recent developmental process and future research direction **7**(12) (2017). https://doi.org/10.3390/app7121211, doi:10.3390/app7121211

10. Imène, S., Badia, K., Nadir, M.: Knowledge graph-based approaches for related entities recommendation **361 LNNS**, 488 – 496 (2022). https://doi.org/10.1007/978-3-030-92038-8_49, doi:10.1007/978-3-030-92038-8_49
11. Kumar, D.: Goodreads book data. `https://www.kaggle.com/datasets/deepaktheanalyst/books-details-dataset` (2022), accessed: April 12, 2025
12. Melville, P., Mooney, R.J., Nagarajan, R.: Content-boosted collaborative filtering for improved recommendations. In: Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02). pp. 187–192. Edmonton, Alberta (2002), `http://www.cs.utexas.edu/users/ai-lab?melville:aaai02`
13. Nilashi, M., Ibrahim, O., Bagherifard, K.: A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques **92**, 507 – 520 (2018). https://doi.org/10.1016/j.eswa.2017.09.058, doi:10.1016/j.eswa.2017.09.058
14. Roy, D., Dutta, M.: A systematic review and research perspective on recommender systems **9**(1) (2022). https://doi.org/10.1186/s40537-022-00592-5, doi:10.1186/s40537-022-00592-5
15. Sun, Y., Mi, J., Jin, C.: Entropy-based concept drift detection in information systems **290** (2024). https://doi.org/10.1016/j.knosys.2024.111596, doi:10.1016/j.knosys.2024.111596
16. Yan, K.: Optimizing an english text reading recommendation model by integrating collaborative filtering algorithm and fasttext classification method **10**(9) (2024). https://doi.org/10.1016/j.heliyon.2024.e30413, doi:10.1016/j.heliyon.2024.e30413
17. Zangerle, E., Bauer, C.: Evaluating recommender systems: Survey and framework **55**(8) (2023). https://doi.org/10.1145/3556536, doi:10.1145/3556536