IIT.SRC 2025 Evaluating Recommender Systems for Digital Library Datasets

Ákos Lévárdy*

Faculty of Informatics and Information Technologies STU in Bratislava
{abc,lncs}@uni-heidelberg.de

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, and FastText 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. By benchmarking these factors, we highlight the trade-offs between accuracy and computational efficiency, offering insights into the strengths and weaknesses of each algorithm for book recommendation tasks in digital 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].

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

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

analyzing large volumes of textual data, they aim to understand users' preferences and generate personalized recommendations [15].

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.

It's also important to recognize the dual perspective from which recommendation systems operate [12]. On one hand, service providers aim to use these systems to sell more items, improve user satisfaction, or better understand customer behavior. On the other hand, from the user's viewpoint, RS assist in discovering suitable items, recommending sequences, or even influencing others' decisions.

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 [16]. 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 [14]. 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 [13]. 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).

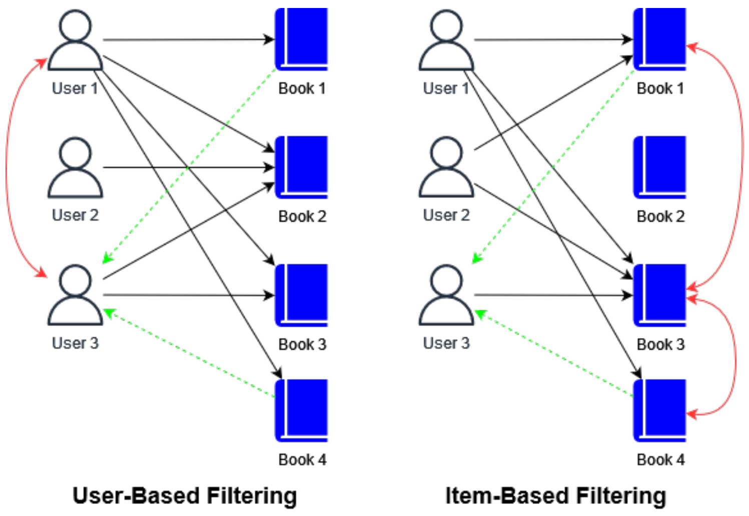

Fig. 1. Illustration of Memory-based CF recommendation.

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. 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.
Popularity-Based recommendations offer products that are popular or well-liked by a lot of users. They assume that these popular items are likely to be of interest to the majority of users, not considering their personal preferences.
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.

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].

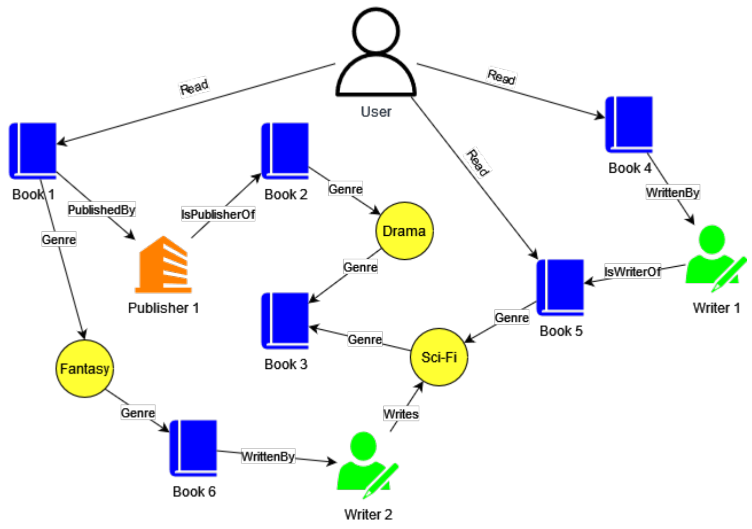

Fig. 2. Illustration of KG-aware recommendation.

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 collaborative 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 [11].

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, meaning their whole text and summaries were extracted from the digital library and separeted into JSON files, each file contained one books whole data, which was even more separated into sentences, pages and paragraphs.
From extracting metadata of the books from the digital library, the dataset had 403 different books short summaries that were used in the testing phase. On the other hand, the full text extraction resulted in a dataset of 45 163 paragraphs where the average paragraph word count was around 63.73 words and the longest paragraph had 2016 words.
Each algorithm began by embedding or vectorizing all of the paragraphs in the dataset according to its specific requirements. This process involved transforming the text into numerical representations, such as vectors, which could then be compared based on their similarities. After the entire dataset of paragraphs was embedded or vectorized, the system received an input paragraph. This input paragraph was also embedded or vectorized in the same manner as the rest of the dataset. Once the input paragraph was transformed, the system calculated the cosine similarity between the input paragraph's vector and the vectors of each of the other paragraphs in the dataset.

The cosine similarity measure was used to determine how closely related the input paragraph was to each of the other paragraphs. Cosine similarity is a mathematical measure that assesses the angle between two vectors, with a smaller angle indicating higher similarity. Based on the computed cosine similarity scores, the paragraphs in the dataset were sorted, with the most similar ones ranked at the top. From this sorted list, the top 5 most similar paragraphs were selected as the algorithm's recommendations for the input paragraph. These top 5 paragraphs were returned to the user as the most relevant or similar ones based on the cosine similarity metric.

Descriptions of the Algorithms Used
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. Below is an overview of common algorithms, highlighting their methodology and key characteristics:

TF-IDF (Term Frequency - Inverse Document Frequency) creates two matrices that are interrelated, trying to figure out the relevancy of a given term (word) to a document given a larger body of documents.

TF means how often a given word occurs in the given document, because words that occur frequently are probably more important. DF means how often the given word occurs in an entire set of documents, but this does not have to mean the word is important, it just shows common words that appear everywhere. So using Inverse DF shows how often the word appears in a document, over how often it appears everywhere.

BoW (Bag of Words) creates a set of vectors containing the count of word occurrences in the document. Unlike TF-IDF, BoW just counts the occurrences of unique words and puts them in its vocabulary so each word becomes a feature or dimension. Each document is represented as a vector based on the frequency of words from the vocabulary. The term-document matrix represents the documents as rows and the unique words as columns with cells showing frequency.

GloVe (Global Vectors for Word Representation) is a word embedding model that builds a co-occurrence matrix where rows represent the words, columns represent the context words and each cell contains the frequency with which the word and context word co-occur within a specified window. The matrix is factorized to learn word embeddings. After training GloVe produces embeddings for all words in the vocabulary.

LSA (Latent Semantic Analysis) is a model for extracting and representing the contextual-usage meaning of words by statistical computations applied to a large corpus of documents. It uses Singular Value Decomposition (SVD) and Rank lowering (Dimensionality reduction). The SVD splits the Matrix of document by keyword into three matrices, which are topic by keyword, document by topic and the diagonal matrix.

FastText breaks words into character n-grams and learns embeddings for these subwords. It can handle words that are out of the vocabulary, because it models character n-grams and not words. For output it produces dense, fixed-length word vectors that have the additional subword information.

Table 1. Results of Different Algorithms - Using Paragraphs

Algorithm	**Avg Similarity**	**Avg Confidence**	**Avg Diversity**	**Avg Elapsed_time (s)**
TF-IDF	0.41	0.04	0.59	13.86
BoW	0.50	0.13	0.50	13.25
FastText	0.98	0.00	0.02	42.81
GloVe	0.99	0.00	0.01	14.39
LSA	0.66	0.02	0.34	12.66

3 Experimental Results

The experiment results from the two tables present the performance of different algorithms evaluated based on two distinct content types: paragraphs and

Table 2. Results of Different Algorithms - Using Summaries

Algorithm	**Avg Similarity**	**Avg Confidence**	**Avg Diversity**	**Avg Coverage (%)**
TF-IDF	0.18	0.092	0.82	3.28
BoW	0.256	0.108	0.744	4.14
FastText	0.993	0.005	0.007	37.44
GloVe	0.983	0.021	0.017	8.14
LSA	0.497	0.133	0.503	4.17

summaries. The metrics used to evaluate the algorithms include average similarity, confidence, diversity, and coverage. Here's a brief description of these metrics: **Average similarity** indicates how closely the recommendations align with the input content, with higher values representing more relevant recommendations. **Confidence** measures the algorithm's certainty about the relevance of its recommendations. Higher values suggest the algorithm is more confident in its suggestions. **Diversity** reflects the variety of recommendations made. A higher diversity value indicates that the algorithm provides a broader range of suggestions. **Coverage** refers to the percentage of items from the dataset that the algorithm is able to recommend. A higher coverage indicates that the algorithm suggests a larger portion of the available items.

Performance on Paragraphs is shown in the first table, which contain more detailed information compared to summaries. As such, the metrics for paragraphs may show higher values for relevance and coverage, since longer text provides more context.

FastText stands out with the highest **average similarity** of 0.98, meaning its recommendations are highly relevant. However, this comes at the cost of **average elapsed time** (42.81 seconds), indicating that it is slower compared to other algorithms. It performs well in terms of relevance but is slower in processing. **TF-IDF** and **BoW**, while being faster (13.25 and 13.86 seconds for **avg_elapsed_time**), show lower **average similarity** (0.41 and 0.50, respectively), meaning their recommendations are less aligned with the user's preferences. These algorithms are quicker but less accurate. **GloVe** and **LSA** perform moderately well in terms of similarity, with **LSA** being slightly faster than **GloVe**.
Overall, the first table highlights the trade-off between accuracy and efficiency: algorithms like **FastText** provide more accurate results but take more time to process, while others like **TF-IDF** and **BoW** are quicker but less precise in their recommendations.

Performance on Summaries is shown in the second table. Since summaries are shorter and less detailed than paragraphs, the results show lower accuracy (i.e., **average similarity**) but provide a broader view of how the algorithms behave with concise content. The coverage of each algorithm was set to return recommended items above the same dynamically set threshold.

FastText again performs the best in **average similarity** with a value of 0.993, which indicates its ability to generate highly relevant recommendations. It also has the highest **coverage** at 37.44%, suggesting it can recommend a large portion of the dataset. However, its **confidence** and **diversity** values are lower, suggesting that while the recommendations are relevant, they may be more focused and less diverse. **GloVe** performs well with an **average similarity** of 0.983, meaning its recommendations are quite relevant, though its **coverage** is only 8.14%, implying it covers fewer items from the dataset compared to algorithms like **FastText**. **LSA** shows a moderate **average similarity** of 0.497, indicating that its recommendations are less aligned with the summaries compared to **FastText** and **GloVe**. However, it strikes a good balance between **diversity** (0.503) and **coverage** (4.17%), suggesting that **LSA** provides more varied recommendations but with less precision in terms of relevance. **TF-IDF** has the lowest **average similarity** value of 0.18, indicating that its recommendations are not closely aligned with the user's preferences. However, it performs well in terms of **diversity** (0.82), suggesting that it provides a variety of recommendations, although these are less relevant. Its **coverage** is low at 3.28%, meaning it can only recommend a small portion of the available dataset. Overall, the table illustrates the trade-off between relevance, diversity, and coverage. Algorithms like **FastText** and **GloVe** provide more relevant recommendations but vary in their coverage of the dataset. **TF-IDF** provides more diversity but less relevance, while **LSA** offers a balance between diversity and coverage at the expense of similarity. These metrics allow us to assess the strengths and weaknesses of each algorithm in terms of their ability to meet the requirements of a recommendation system.

4 Future Work

Future work could explore improving the efficiency of the algorithms by implementing more advanced optimization techniques to reduce the computational time and memory usage, especially for models like FastText. Additionally, hybrid approaches combining content-based filtering with collaborative filtering could be tested to enhance the accuracy of recommendations by considering both item features and user preferences. Further evaluation of these algorithms could involve larger datasets and more diverse types of content to assess the scalability and generalization of the models. Exploring deep learning-based methods, such as using transformer models or neural networks, could also be an interesting direction for improving recommendation quality. Finally, incorporating user feedback to dynamically adjust recommendations in real-time could be an important area of focus.

5 Conclusions

This study focused on evaluating content-based recommender systems for digital library datasets, comparing several algorithms such as TF-IDF, LSA, FastText, GloVe, and BoW. The goal was to assess how well these algorithms can generate recommendations based on the textual content of books and their paragraphs, and how efficiently they perform under different computational loads.
From the results of the performance metrics, it is evident that there are trade-offs between relevance, diversity, and efficiency. Algorithms like FastText and GloVe offer highly relevant recommendations but tend to require more computational resources in terms of processing time. On the other hand, methods such as TF-IDF and BoW are faster but offer lower similarity, meaning the recommendations are less aligned with the user's preferences. LSA provides a balance between similarity and diversity, though it still lags behind the best-performing models in terms of relevance.

Additionally, the analysis of coverage showed how well the algorithms could recommend a wide range of items from the dataset. FastText, while highly relevant, exhibited lower diversity and coverage compared to GloVe, indicating that its recommendations are more focused. In contrast, TF-IDF offered higher diversity but lower relevance, meaning it could recommend a wider variety of items. The findings suggest that hybrid models combining content-based filtering with collaborative filtering techniques could improve recommendation quality, as they would leverage both textual content and user behavior data. Further research could explore deep learning models, such as neural networks, to detect more complex patterns in large datasets and improve recommendations.
In conclusion, this research provides valuable insights into the strengths and weaknesses of various content-based algorithms for digital libraries, offering a framework to optimize recommendation systems.

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, `https://app.dimensions.ai/details/publication/pub.1034486657`, 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, `https://app.dimensions.ai/details/publication/pub.1022525812`, 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, `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85121759193&doi=10.1007%2f978-3-030-92038-8_92&partnerID=40&md5=133d7db406b9825456059a5d0cefdca7`, 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, https://www.scopus.com/inward/record.uri?eid=2-s2.0-84891925287&doi=10.1007%2f978-3-642-41338-4_3&partnerID=40&md5=7ffac1eca772e767fa2f053a6bc67060, 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, https://app.dimensions.ai/details/publication/pub.1072601078, 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, https://www.scopus.com/inward/record.uri?eid=2-s2.0-0036959356&doi=10.1023%2fA%3a1021240730564&partnerID=40&md5=b3badfce58ee2788c6422ab1a6c1bf4e, 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, https://www.scopus.com/inward/record.uri?eid=2-s2.0-84923290092&doi=10.1016%2fj.procs.2014.10.015&partnerID=40&md5=3568e689c7fc885bc1dc4372f16a3324, 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, https://www.scopus.com/inward/record.uri?eid=2-s2.0-85150943074&doi=10.1007%2f978-1-0716-2197-4_15&partnerID=40&md5=4c9d05ad01fa1fd92d7264f12b2b2015, 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, https://www.scopus.com/inward/record.uri?eid=2-s2.0-85037151889&doi=10.3390%2fapp7121211&partnerID=40&md5=97216c007e2be2e0069af24c05b82ebf, 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, https://www.scopus.com/inward/record.uri?eid=2-s2.0-85121688950&doi=10.1007%2f978-3-030-92038-8_49&partnerID=40&md5=cba7c2bee5ca48e5a30eda4000c33ef3, doi:10.1007/978-3-030-92038-8_49
11. 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
12. Ricci, F., Rokach, L., Shapira, B.: Recommender Systems: Techniques, Applications, and Challenges (2022). https://doi.org/10.1007/978-1-0716-2197-4_1, https://www.scopus.com/inward/record.uri?eid=2-s2.0-85160177777&doi=10.1007%2f978-1-0716-2197-4_1&partnerID=40&md5=91db3bd9df8d56dede2fb845e8b4cbbd, doi:10.1007/978-1-0716-2197-4_1
13. 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, https://www.scopus.com/inward/record.uri?eid=2-s2.0-

85129497479&doi=10.1186%2fs40537-022-00592-5&partnerID=40&md5=0ec10720a5635425eb3d18626cb13f13, doi:10.1186/s40537-022-00592-5
14. 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
15. 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, https://www.scopus.com/inward/record.uri?eid=2-s2.0-85191347340&doi=10.1016%2fj.heliyon.2024.e30413&partnerID=40&md5=d5290bf49882909419bc5234453ee206, doi:10.1016/j.heliyon.2024.e30413
16. Zangerle, E., Bauer, C.: Evaluating recommender systems: Survey and framework **55**(8) (2023). https://doi.org/10.1145/3556536, https://www.scopus.com/inward/record.uri?eid=2-s2.0-85177779269&doi=10.1145%2f3556536&partnerID=40&md5=c105ead68d5ac878830699c16e107228, doi:10.1145/3556536