Evaluating Recommender Systems for Digital Library Datasets

Ákos Lévárdy

ABSTRACT
With the increasing amount of digital content online, recommender systems play an important role in filtering out and helping users navigate large information spaces by generating personalized recommendations. This study focuses on evaluating content-based recommender systems, which are specifically designed for digital library datasets. These content-based filtering methods analyze textual data from books to generate recommendations based on similarity between content features. The goal is to investigate different algorithms such as TF-IDF, LSA, GloVe or FastText, and testing them with different settings to assess their effectiveness in generating Top-N recommendations from book text data. In order to fully evaluate these algorithms, we conduct offline experiments using multiple performance metrics like similarity, diversity, confidence or even coverage. Additionally, the performance of the algorithms is analyzed by tracking execution time, CPU usage, and memory consumption. By benchmarking these factors, we provide information on the trade-offs between accuracy and computational efficiency for the different tested models. Digital libraries can improve user experience by selecting the most effective and suitable recommendation systems. This research identifies the strengths and weaknesses of the selected algorithms for book recommendations, making it easier to navigate large digital collections.

KEYWORDS:
Recommender System, Content-Based, Evaluation, Performance Metrics, Digital Library

INTRODUCTION

As the internet and technology continue to evolve rapidly, the amount of information available online has grown significantly, covering areas like online shopping, government services, entertainment, education, and much more. With so much information out there, it's easy for users to feel overwhelmed. This is where Recommender Systems (RS) come in. These systems are essential for improving the user experience by helping people find the most relevant information, saving time and effort. Rather than showing everything available, they predict what a user is most likely to be interested in, offering suggestions based on their needs and preferences. Recommender systems are powerful tools that make it easier for users to find content that suits their tastes, whether it's a book, movie, product, or music.
The main idea behind recommender systems is to personalize the content shown to each user [1]. By using algorithms that analyze past behavior and preferences, these systems suggest items that align with a person's individual interests. For example, in a digital library, a recommender system suggests books that might be of interest based on the user's reading history.
The design of these systems varies depending on the type of content being recommended, whether it's movies, books, or any other type of media. A movie recommendation system would work differently from a book recommender system because of the specific characteristics [2].
There are different methods that recommender systems use to predict what a user might like. Collaborative filtering looks at the behavior of similar users and suggests items based on what others with similar tastes have liked. Content-based filtering, on the other hand, analyzes the features of the items themselves, such as genre, author, or keywords, and recommends similar items based on what the user has liked before. Some systems even use a mix of both approaches, known as Hybrid methods, to make better and more accurate suggestions. Regardless of the method, the main goal is to help users find content that is most relevant to them by under-

standing the connections between what they like and what's available.
At the core of these systems is the idea that there are relationships between what users are interested in and the items they interact with. For instance, if a user watches a historical documentary, they are more likely to enjoy another documentary on a similar topic, like ancient history, than a completely different genre, like action movies [3]. Recommender systems aim to identify these relationships and help users discover content they might not have come across on their own.
The focus of this paper was on evaluating different algorithms and comparing their performance and results. This evaluating system is specifically set for recommending books and parts or paragraphs of these books based on their textual content. With the vast number of books available, it's important to have an efficient system that can help users find books they will enjoy [4]. The system will test different algorithms, including TF-IDF, LSA, BERT and so on which are all designed to analyze text and make recommendations. Each of these algorithms works differently, so it's important to compare them and see how well they perform in recommending books and paragraphs.
To evaluate the effectiveness of these algorithms, the system will be tested based on several factors, like similarity, diversity, confidence and coverage [5]. These metrics help measure how well the system recommends books that match the user's interests, how varied the suggestions are, and how much of the available content is covered. Additionally, the project will also look at how efficient the system is by monitoring execution time, memory usage, and CPU performance. This allows us to balance both accuracy and efficiency, ensuring the system works well without overloading the computer.
By testing these algorithms and considering both their effectiveness and efficiency, this project aims to provide a useful tool for digital libraries, helping users find the right books while improving their overall experience. Ultimately, the goal is to make the book discovery process easier and more enjoyable, so users can explore large collections without feeling lost.

UNDERSTANDING RECOMMENDATION SYSTEMS

Making decisions is not always easy. People are frequently presented with an overwhelming number of options when picking a product, a movie, or a destination to travel to, and each option comes with different levels of information and trustworthiness.
While there are many situations in which users know exactly what they are looking for and would like immediate answers, in other cases they are willing to explore and extend their knowledge [6].

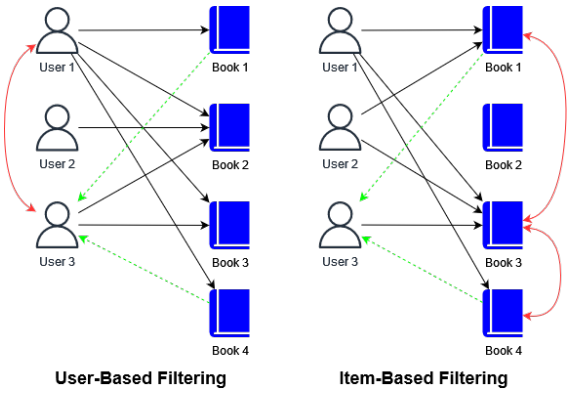
The main purpose of **Recommendation Systems** is to predict useful items, select some of them and after comparing them, the system recommends the most accurate ones.
These Personalized recommendation systems are emerging as appropriate tools to aid and speed up the process of information seeking, considering the dramatic increase in big data [7]. They need to handle a large amount of textual data in order to accurately understand users' reading preferences and generate corresponding recommendations [8].
Because of this number of detail from all of the items, recommendation systems are becoming increasingly important. They help reduce options and offer better suggestions for the user so that they will have a personalized list to select their favourite. Fast and efficient access to information is essential in any field of study.

While both Recommendation Systems (Information Filtering techniques) and Search Engines (Information Retrieval techniques) aim to help users navigate all this information, they do it differently. Personalized recommendation systems make suggestions based on past user behavior and preferences, whereas search engines use keyword-based searches to retrieve content from a selection of sources [9].
Information systems often deal with changing data over time. The term called Concept drift describes when sometimes the patterns or behaviors in the data change unexpectedly which affects how the system makes predictions [10].
It is also crucial to distinguish on whose behalf the role of the recommendation system is played. One role could be the service providers role, who would like to sell more items, increase user satisfaction or better understand what the user wants with recommendations [11]. The other role is from the users point of view the recommendation system can be helpful with tasks such as finding some good items, recommending a sequence or influencing others to consider particular products.
The task to provide users with currently available options for products that fit their requirements and interests is very important in todays consumer society. These products are mostly supplied by inputs [12] , sometimes even matching the users distinct tastes.
When someone is trying to find a movie to watch, it would be hard for them to start searching with-

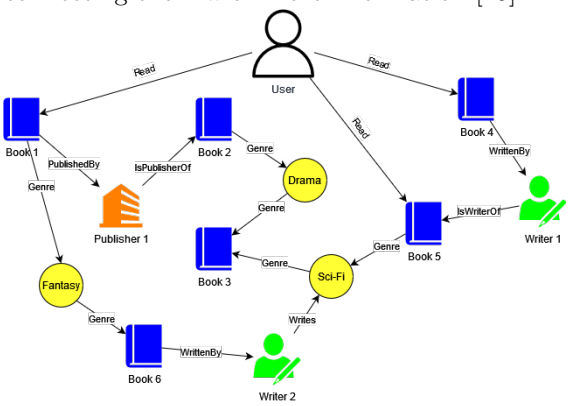
out any starting options. After all a blank page and no suggestions to choose from might even make the user decide not to pick anything.

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) [14].
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).



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 [15].



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 [16].
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 [17].

The most popular techniques used are the Collaborative Filtering, Content-Based Filtering and the Hybrid method [18]. 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.

SYSTEM DESCRIPTION

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.

Algorithm	Avg Similarity	Avg Confidence	Avg Diversity	Avg Elapsed_time (s)	Avg IPS
TF-IDF	0.41	0.04	0.59	13.86	3259.31
BoW	0.50	0.13	0.50	13.25	3404.92
FastText	0.98	0.00	0.02	42.81	1055.02
GloVe	0.99	0.00	0.01	14.39	3139.40
LSA	0.66	0.02	0.34	12.66	3567.57

Table 1: Results of Different Algorithms - Using Paragraphs

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

Table 2: Results of Different Algorithms - Using Summaries

EXPERIMENTAL RESULTS

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

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.

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.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor for their invaluable guidance and support throughout the duration of this project.

References

[1] Ali Taleb Mohammed Aymen and Saidi Imène. Scientific paper recommender systems: A review. 361 LNNS:896 – 906, 2022. doi:10.1007/978-3-030-92038-8_92.

[2] *Introduction to Recommender Systems Handbook*, pages 1–35. 2010. doi:10.1007/978-0-387-85820-3_1.

[3] Charu C. Aggarwal. *An Introduction to Recommender Systems*. 2016. doi:10.1007/978-3-319-29659-3.

[4] Eva Zangerle and Christine Bauer. Evaluating recommender systems: Survey and framework. 55(8), 2023. doi:10.1145/3556536.

[5] Asela Gunawardana, Guy Shani, and Sivan Yogev. *Evaluating Recommender Systems*. 2022. doi:10.1007/978-1-0716-2197-4_15.

[6] Roi Blanco, Berkant Barla Cambazoglu, Peter Mika, and Nicolas Torzec. Entity recommendations in web search. 8219 LNCS(PART 2):33 – 48, 2013. doi:10.1007/978-3-642-41338-4_3.

[7] Khalid Haruna, Maizatul Akmar Ismail, Suhendroyono Suhendroyono, Damiasih Damiasih, Adi Cilik Pierewan, Haruna Chiroma, and Tutut Herawan. Context-aware recommender system: A review of recent developmental process and future research direction. 7(12), 2017. doi:10.3390/app7121211.

[8] Ke Yan. Optimizing an english text reading recommendation model by integrating collaborative filtering algorithm and fasttext classification method. 10(9), 2024. doi:10.1016/j.heliyon.2024.e30413.

[9] D. De Nart and C. Tasso. A personalized concept-driven recommender system for scientific libraries. volume 38, page 84 – 91, 2014. doi:10.1016/j.procs.2014.10.015.

[10] Yingying Sun, Jusheng Mi, and Chenxia Jin. Entropy-based concept drift detection in information systems. 290, 2024. doi:10.1016/j.knosys.2024.111596.

[11] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems: Techniques, Applications, and Challenges*. 2022. doi:10.1007/978-1-0716-2197-4_1.

[12] Simon Philip, P.B. Shola, and Abari Ovye John. Application of content-based approach in research paper recommendation system for a digital library. *International Journal of Advanced Computer Science and Applications*, 5(10), 2014. doi:10.14569/IJACSA.2014.051006.

[13] Deepjyoti Roy and Mala Dutta. A systematic review and research perspective on recommender systems. 9(1), 2022. doi:10.1186/s40537-022-00592-5.

[14] *Content-based Recommender Systems: State of the Art and Trends*, pages 73–105. 2010. doi:10.1007/978-0-387-85820-3_3.

[15] Saidi Imène, Klouche Badia, and Mahammed Nadir. Knowledge graph-based approaches for related entities recommendation. 361 LNNS:488 – 496, 2022. doi:10.1007/978-3-030-92038-8_49.

[16] Robin Burke. Hybrid recommender systems: Survey and experiments. 12(4):331 – 370, 2002. doi:10.1023/A:1021240730564.

[17] Prem Melville, Raymond J. Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, pages 187–192, Edmonton, Alberta, 2002.

[18] Poonam B.Thorat, R. M. Goudar, and Sunita Barve. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110(4):31–36, 2015. doi:10.5120/19308-0760.