Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
FIIT-16768-121314

Ákos Lévárdy

Evaluating Recommender Systems for Digital Library Datasets

Bachelor's thesis

Study programme: Informatics
Study field: Computer Science
Place: Institute of Informatics, Information Systems and Software Engineering
Supervisor: PaedDr. Pavol Baťalík

May 2025

Slovak University of Technology in Bratislava
Institute of Informatics, Information Systems and Software Engineering

Faculty of Informatics and Information Technologies
Academic year: 2024/2025
Reg. No.: FIIT-16768-121314

BACHELOR THESIS TOPIC

Student: **Ákos Lévárdy**
Student's ID: 121314
Study programme: Informatics
Study field: Computer Science
Thesis supervisor: PaedDr. Pavol Baťalík
Head of department: doc. Ing. Ján Lang, PhD.

Topic: **Evaluating Recommender Systems for Digital Library Datasets**

Language of thesis: English

Specification of Assignment:

Byť vždy včas a správne informovaný veľkou mierou pomáha pri našom rozhodovaní. Nájsť dostatočné informácie pre vyriešenie úloh a problémov, s ktorými sa stretávame a rýchlo a bezpečne sa v nich zorientovať je nevyhnutnosťou v každodennom živote, ako aj pri štúdiu. Systém, ktorý by dokázal počas vyhľadávania doporučovať relevantné informácie súvisiace s predmetom nášho štúdia a tým výrazne napomáhal v získavaní nových pre nás užitočných poznatkov. Zmapujte aktuálne využívané webové odporučiace (recommendation) systémy a algoritmy. Tiež možnosti obdobných nástrojov na báze AI. Analyzujte možnosti ich využitia pre odporúčacie systémy v kontexte digitálnych knižníc a repozitárov. Navrhnite a implementujte vlastné riešenie využitia odporúčacích systémov pre zjednodušenie vyhľadávania informácií. Vyhodnoťte implementované riešenie nad reálnymi dátami z voľne dostupného repozitára.

Length of thesis: 40

Deadline for submission of Bachelor thesis: 12. 05. 2025
Approval of assignment of Bachelor thesis: 15. 04. 2025
Assignment of Bachelor thesis approved by: doc. Ing. Ján Lang, PhD. – Study programme supervisor

Declaration of honour
I declare on my honour that I wrote this thesis single-handed with usage of quoted literature and based on my knowledge and professional supervision of my supervisor PaedDr. Pavol Baťalík.

In Bratislava, 12.05.2025

Ákos Lévárdy

Acknowledgement
First and foremost, I would like to thank my supervisor for their invaluable guidance and support throughout the duration of this project. I would also like to thank Ing. Jakub Dubec, whose assistance and feedback were greatly appreciated.

Annotation

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: Informatics

Author: Ákos Lévárdy
Bachelor's Thesis: Evaluating Recommender Systems for Digital Library Datasets
Supervisor: PaedDr. Pavol Baťalík
2025, May

This bachelor thesis focuses on analysing and evaluating recommender systems best suitable for digital library datasets. With the rise of digital content, such systems help users navigate large information spaces by generating personalized recommendations. In the introduction we show the importance and role of Recommender Systems (RS) in filtering and anticipating user preferences across various domains. We then present an overview of the key recommendation techniques, explaining their underlying mechanisms and common challenges. The project emphasizes Content-Based Filtering techniques, comparing text representational methods like BERT, FastText, LSA, and TF-IDF. Offline experiments were conducted to evaluate the system's ability to generate Top-N book recommendations from the input data and the algorithms were compared based on evaluation metrics such as similarity, coverage, diversity or confidence. The main objective of this work aims to design and implement a system capable of benchmarking different recommendation algorithms designed for digital library datasets and to provide insights into their effectiveness and applicability in this domain.

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Informatika

Autor: Ákos Lévárdy
Bakalárska práca: Vyhodnocovanie odporúčacích systémov pre datasety digitálnych knižníc
Vedúci bakalárskej práce: PaedDr. Pavol Baťalík
Máj, 2025

Táto bakalárska práca sa zameriava na analýzu a hodnotenie odporúčacích systémov, ktoré sú najvhodnejšie pre datasety digitálnych knižníc. S rastom digitálneho obsahu tieto systémy pomáhajú používateľom orientovať sa v rozsiahlych informačných priestoroch generovaním personalizovaných odporúčaní. V úvode je predstavený význam a úloha odporúčacích systémov pri filtrovaní a predpovedaní používateľských preferencií v rôznych oblastiach. Práca následne poskytuje prehľad kľúčových odporúčacích techník, pričom vysvetľuje ich základné princípy a bežné problémy. Projekt kladie dôraz na techniky založené na obsahovom filtrovaní, pričom porovnáva rôzne prístupy na reprezentáciu textu, ako BERT, FastText, LSA alebo TF-IDF. Boli vykonané offline experimenty na vyhodnotenie schopnosti systému generovať odporúčania Top-N kníh na základe vstupných dát. Algoritmy boli testované a porovnávané podľa hodnotiacich metrík ako napríklad podobnosť, pokrytie, diverzita alebo dôvera. Hlavným cieľom tejto práce je navrhnúť a implementovať systém schopný porovnávať rôzne odporúčacie algoritmy určené pre digitálne knižničné dáta a poskytnúť prehľad o ich efektívnosti a použiteľnosti v danej oblasti.

Table of Contents

List of Figures

List of Tables

1 Introduction

As Internet and Web technologies continue to evolve rapidly, the amount of information available online has expanded excessively across sections such as e-commerce, e-government or e-learning. To help users navigate this vast sea of content, Recommender Systems (RS) have become fundamental. These systems are not designed just for saving time for users, but they are enhancing the users experience when using the said system, by anticipating their needs and relevant items or topics to discover. They are very effective tools for filtering out the most appropriate information any user would like to find. The primary focus of these recommendations is to predict if a specific user will be interested in the distinct items.
"Item" is the general term used to refer to what the system recommends to users. A RS normally focuses on a specific type of item (e.g., movies, books or news) and accordingly its design, its graphical user interface, and the core recommendation technique used to generate the recommendations are all customized to provide useful and effective suggestions for that specific type of item. [4]
The basic principle of recommendations is that significant dependencies exist between user- and item-centric activity. For example, a user who is interested in a historical documentary is more likely to be interested in another historical documentary or an educational program, rather than in an action movie. [5]
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 target of this project is to create a recommendation system that uses different algorithms in the topic of book recommendations and evaluate these algorithms based on specified metrics.

2 Understanding Recommendation Systems

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].
Similarly, **Search Engines** are essential for navigating the vast amount of information available online. They make it possible for people to quickly look up solutions, learn new things, and browse the wide variety of resources on the internet. Search engine optimization is now necessary to guarantee that search engines deliver relevant results, quick search times, and a top-notch user experience given the explosive growth of online information.
A search engine is essentially a software that finds the information the user needs using keywords or phrases. It delivers results rapidly, even with millions of websites available online. The importance of speed in online searches is highlighted by how even minor delays in retrieval can negatively affect users' perception of result quality [9].
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.
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].
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 [11] , 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 without 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. Below are different recommendation systems listed.

Basic ideas of the recommendation techniques:

- **Content-Based Filtering** works in a way that it creates user profiles and suggests the individual items or products based on the users past choices with similar items. The items have various features and characteristics which connect them [12].
- **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 [13].
- **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 and detail [14].
- **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 [7]. 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 [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 [15].

– **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 [16].

2.1 Role of Recommendation Systems

The Recommendation System can have a range of roles to play. First it is important to distinguish on whose behalf the role is played, which can be either the service providers or the users side. For example, a recommendation system for music, implemented by a streaming service like Spotify wants to increase user engagement by recommending new playlists and songs, which leads to more subscriptions or advertisement revenue. While on the other hand the user wants to listen to personalized playlists and discover songs they might like.
There are more ways why a service provider would want to utilize such technology: [4]

- Sell more items - to be able to sell additional items beyond those which are normally sold without recommendations. To increase the number of users that accept a recommendation and consume an item.
- Sell more diverse items - not just to sell the most popular items, but also recommend items that might be hard to find. The popular items will probably be sold either way, on the other hand the service provider might want to sell every item.
- Increase user satisfaction - improve the experience for the user with effective recommendations and combine it with a usable interface, so the user will be more satisfied with the system.
- Increase user fidelity - make more personalized recommendations based on the users previous visits and interactions, by treating the user as a valuable customer.
- Better understand what the user wants - to describe the user's prefer-

ences which are explicitly collected or predicted by the system. The service provider can even use this knowledge for other goals like improve inventory management or target specific promotions.

From the users point of view the recommendation system can help in implementing other core tasks which are normally associated with an RS. The popular tasks are the following: [17]

1. Find some good items - Identify a selection of quality items.
2. Find all good items - Locate all available items deemed good.
3. Annotate items in context - Emphasize items based on the user's preferences and context.
4. Recommend a sequence - Suggest an order for engaging with items.
5. Recommend a bundle - Propose a set of complementary items together.
6. Just browsing - Explore items without the intention of purchasing.
7. Find credible recommender - Evaluate how effective the system is at making recommendations.
8. Improve the active user's profile - Enhance the system's understanding of the user's preferences.
9. Express self - Share opinions and provide ratings to assist the system.
10. Help others - Evaluate items to guide others in finding what suits them.
11. Influence others - Persuade other users to consider particular products.

2.2 Recommendation Techniques

From a higher perspective recommendation techniques are generaly categorized into three approaches which are Collaborative Filtering, Content-Based Filtering and Hybrid Approaches. These methods differ in how they generate recommendations and offer unique advantages. The efficiency of a recommender system greatly depends on the type of algorithm used and the nature of the data source, which may be contextual, textual, visual etc. [18]
In the following section the techniques Collaborative Filtering, Content-based Filtering, Knowledge Graphs and Hybrid Approaches are described in more detail.

2.2.1 Collaborative Filtering

One of the most popular methods used for personalized recommendations is collaborative filtering (CF). This method filters information from users, which means it compares users behaviour, interactions with items and data, item correlation and ratings from users.
It can perform in domains where there is not much content associated with items, or where the content is difficult for a computer to analyze - ideas, opinions etc. [16]
Collaborative filtering can be divided into 2 methods which are "Memory-based" and "Model-Based" collaborative filtering. The first one relies on historical preferences, whereas the second method is based on machine learning models to predict the best options.

Memory-based CF
Collaborative filtering systems based on memory automate the common principle that similar users prefer similar items, and similar items are preferred by similar users [19].
This method can also be called Neighborhood-based and is further divided into 2 basic types, which are:

- User-Based Collaborative Filtering
  - The main idea is that 2 completely distinct users who have an interest in a specific item and they rate this item similarly will probably be drawn to a new item the same way.

- Item-Based Collaborative Filtering
  - Calculates similarity between items, rather than users. The user will probably like a new item which is similar to another item they were interested in before.

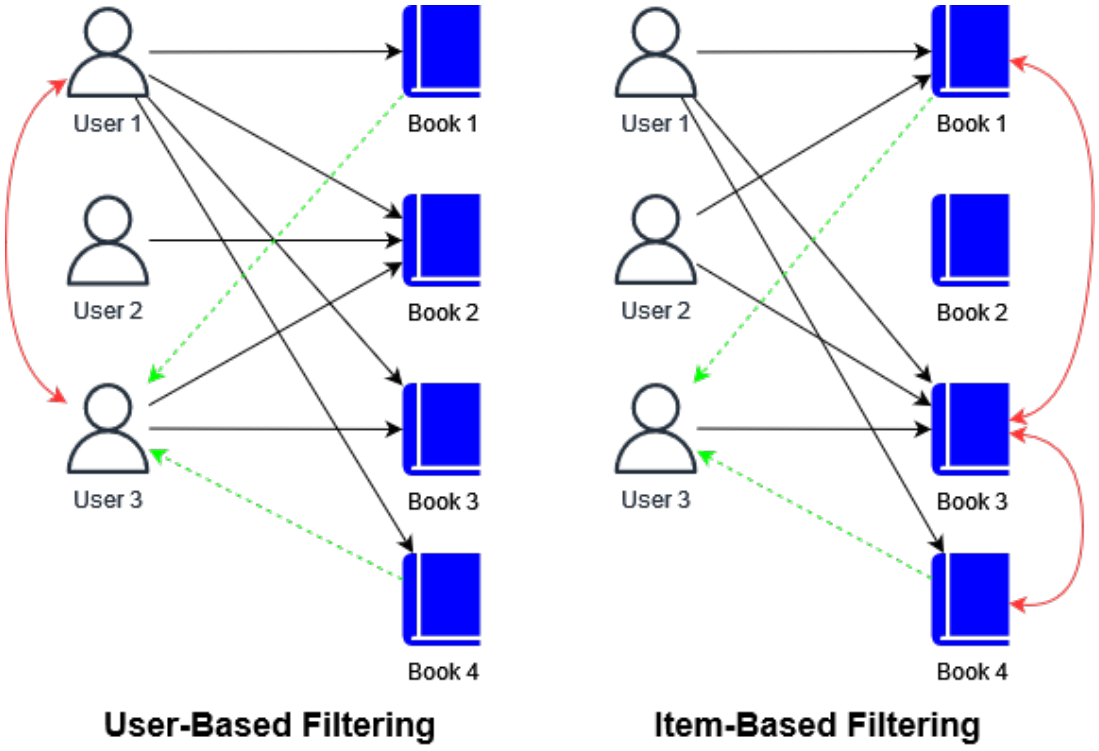

Figure 1: Illustration of Memory-based CF recommendation

When trying to implement this type of recommendation system it is important to consider the key components, which are:

- Rating Normalization - adjusts individual user ratings to a standard scale by addressing personal rating habits. Using for example Mean-Centering or Z-Score Normalization.
- Similarity Weight Computation - helps to select reliable neighbors for prediction and deciding how much impact each neighbor's rating has. A lot of Similarity measures can be used, such as Correlation-Based Similarity, Mean Squared Difference or Spearman Rank Correlation.
- Neighborhood Selection - selects the most appropriate candidates for making predictions based on each unique scenario, eliminating the least likely ones to leave only the best options. [19]

Model-based CF
Collaborative filtering systems based on models, also known as Learning-based methods, try to develop a parametric model of the relationships between items and users. These models can capture patterns in the data, which can not be seen in the previous recommendation type.
Model-based algorithms do not suffer from memory-based drawbacks and can create prediction over a shorter period of time compared to memory-based algorithms because these algorithms perform off-line computation for training. The well-known machine learning techniques for this approach are matrix factorization, clustering and machine learning on the graph [13].

Matrix Factorization
In its basic form, matrix factorization characterizes both items and users by vectors of factors inferred from item rating patterns. High correspondence between item and user factors leads to recommendations [20].
People prefer to rate just a small percentage of items, therefore the user-item rating matrix, that tracks the ratings people assign to various items, is frequently sparse.
In order to deal with this sparsity, matrix factorization (MF) algorithms split the matrix into two lower-rank matrices: one that shows the latent properties of the items and another that reflects the underlying user preferences. These latent representations can be used to predict future ratings or complete the matrix's missing ratings after factorization [21].

It is important to mention that the effectiveness depends on the ratio of users and items. For example when trying to recommend songs, there are usually way more users than songs and generally, many users listened to the same songs or same genres. Which means like-minded users are found easily and the recommendations will be effective. On the other hand, in a different field for example, when recommending books or articles the systems deals with millions of articles but a lot less users. This leads to less ratings on papers or no ratings at all, so it is harder to find people with shared interests [22].

2.2.2 Content-Based Filtering

Recommender Systems which are using content-based filtering, review a variety of items, documents and their details. Each product has their own description which is collected to make a model for each item. The model of an item is composed by a set of features representing its content.
The main benefit of content-based recommendation methods is that they use obvious item features, making it easy to quickly describe why a particular item is being recommended. [12]
This also allows for the possibility of providing explanations that list content features that caused an item to be recommended, potentially giving readers confidence in the system's recommendations and insight into their own preferences [23].
These profiles for items are different representations of information and users interest about the specific item.
The recommendation process basically consists in matching up the attributes of the user profile against the attributes of a content object. [12]
Some additional side information about items can be also useful, where this side information contains additional knowledge about the recommendable items, e.g., in terms of their features, metadata, category assignments, relations to other items, user-provided tags and comments, or related textual content. [24]

The process for recommending items using content-based filtering has 3 different phases and this high level architecture is shown in Fig. 2.:

- Content Analyzer - Turns the unstructured information (text) into structured, organized information using pre-processing steps which are basic methods in Information Retrieval, such as feature extraction.
- Profile Learner - Collects data of the users preference (feedback) that can be either positive information reffering to features which the active user likes or negative ones which the user does not like. After generalization it tries to construct user profiles for later use.
- Filtering Component - Matches the items for the user, based on the similarities between item representations and user profiles, meaning it compares the features of new items with features in user preferences that are stored in the users profile. [12]

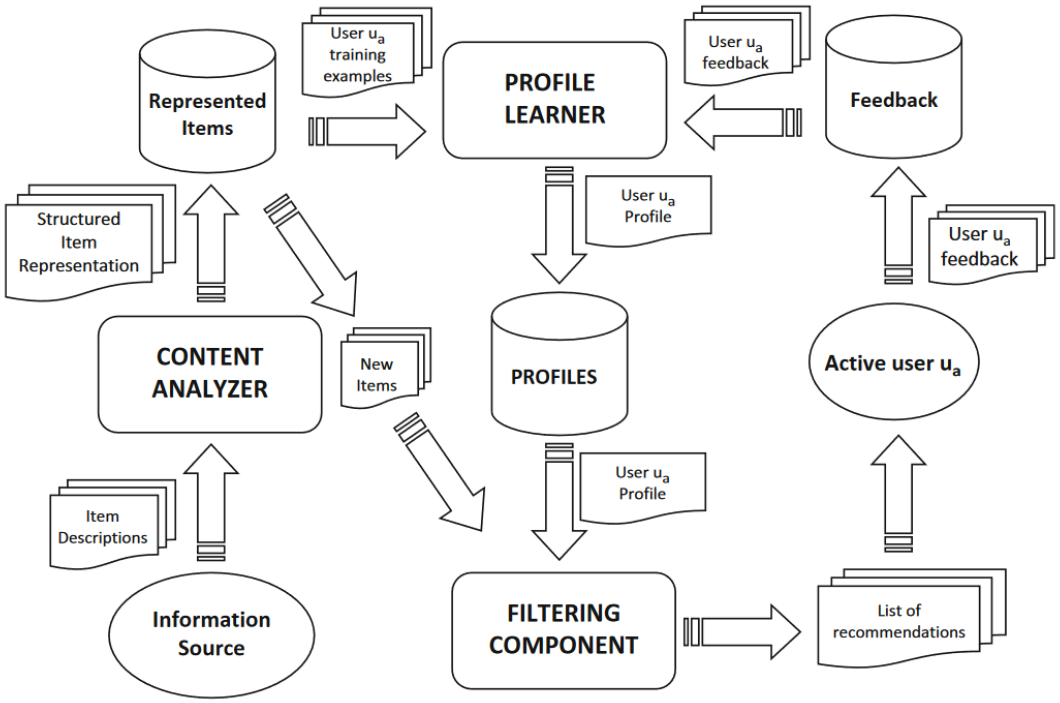

Figure 2: High level architecture of a content-based recommender [1]

The user modeling process has the goal to identify what are the users needs and this can be done 2 ways. Either the system calculates them from the interactions between the user and items through feedback or the user can specify these needs directly by giving keywords to the system, providing search queries [22].

Feedback

When trying to acquire helpful information or criticism that is given by the user there are 2 separate ways. The first one is called Explicit Feedback where it is necessary for the user to give item evaluation or actively rate products. Most popular options are gathering like/dislike ratings on items or the ratings can be on a scale either from 1 to 5 or 1 to 10. After the ratings the user can also give comments on separate items.
The other way is called Implicit Feedback where the information is collected passively from analyzing the users activities. Some alternatives can be clicks on products, time spent on sites or even transaction history [25].

Advantages and Disadvantages of CB Filtering

- User Idependence - meaning the ratings taken into consideration are

only provided by the active users to build their own profiles. Collaborative approaches will recommend items based on feedback from other users in the nearest neighborhood.

- Transparency - explanations for the recommended items can be provided explicitly by listing the content features which were used to get that recommendation. On the other hand Collaborative systems are considered black boxes, where explanations are based on similar tastes of different users.
- New Item - when a new item is added to the system, the Content-based method is capable of recommending it from the set features and its content. This is not possible for the Collaborative method which needs feedback for the new item to be able to recommend it.
- Limited Content Analysis - because the system can only analyze a certain number of features and can miss important aspects such as aesthetics or other multimedia information. Also, systems based on string matching approach can suffer from problems such as synonymy, polysemy or multi-word expressions.
- Over-Specialization - the user will mostly be recommended things similar to what they already liked, which drawback is called 'lack of serendipity'. For example, if the user only rated action movies, then the system would not recommend other genres, which limits the chance of recommending items with novelty or surpise. [25]

Semantic approaches in CB Recommendation

In short, Semantics refer to interpretation of meaning in language, words and symbols. Using semantic techniques the represantations of items and user profiles shift from keyword-based to concept-based ones. With these represantations it is possible to give meaning to information expressed in natural language and to get a deeper understanding of the information presented by textual content.

Content-based recommendations can adopt two different approaches based on how the semantics are derived and applied:

- **Top-Down Semantic Approaches** use an external knowledge to improve the representation of the items and users. This external knowledge can be: ontological resources, encyclopedic knowledge (ESA, Ba-

belNet) and the Linked Open Data cloud. For example the **Ontology** is a structured description that shows how different parts of a system depend on each other and how they are connected. It organizes key concepts in a specific domain into a hierarchy, which can explain their relationships and the characteristics of each concept. It can help to understand how specific examples of these concepts behave and how they are related [26].

- **Bottom-Up Semantic Approaches** use implicit semantic representation of items and user profiles, where the meanings of terms are assumed by analyzing its usage. They rely on the distributional hypothesis: "words that occur in the same context tend to have similar meanings". Without a predefined structure, this approach analyzes the words co-occurrence with other words, larger texts or documents using Discriminative Models. [25]

The semantic approaches can be further categorized by the source of knowledge used to extract meaning, which are **Endogenous Semantics** and **Exogenous Semantics**. In the first case, the semantics is obtained by exploiting unstructured data, and is directly inferred from the available information. Different techniques for these Implicit Semantics Representations are for example the Term Frequency - Inverse Document Frequency (TF-IDF) weighting, or the Distributional Semantics Models (DSM) such as Explicit Semantics Analysis (ESA), Random Indexing or Word Embedding Techniques. Word embedding technology can reflect the semantic information of words to a certain extent. The semantic distance between words can be calculated by word vectors. Commonly used word vectors are based on for example Word2vec or Fasttext models [27].
In the second, the semantics comes from the outside, since it is obtained by mining and exploiting data which are previously encoded in structured and external knowledge sources. For these Explicit Semantics Representations there are also different techniques like Linking Item Features to Concepts using Word Sense Disambiguation (WSD) or using Entity Linking, or Linking Items to Knowledge Graphs using Ontologies or Linked Open Data (LOD). [1]
The LOD cloud is a huge decentralized knowledge base where researchers and organizations publish their data in Resource Description Framework (RDF) format and adopt shared vocabularies, in order to express and interlink the data to each other [28].

2.2.3 Knowledge Graphs

Knowledge graph is a knowledge base that uses a graph-structured data model. It is a graphical database which contains a large amount of relationship information between entities and can be used as a convenient way to enrich users and items information [14].
The idea is that attributes of users and items are not isolated but linked up with each other, which forms a knowledge graph (KG). Incorporating a Knowledge Graph into recommendations can help the results in ways like:

- The rich semantic relatedness among items in a KG can help explore their latent connections and improve the precision of results.
- The various types of relations in a KG are helpful for extending a user's interests reasonably and increasing the diversity of recommended items.
- KG connects a user's historically-liked and recommended items, thereby bringing explainability to recommender systems. [29]

Basically a knowledge graph is a directed graph whose nodes are the entities and the edges are the relations between them. They are usually defined as triplets with a head entity, tail entity and a relationship connecting them. The graphs have detailed supporting information which is background knowledge of items and their relations amongst them. The facts of items are organized in those triplets like (Ed Sheeran, IsSingerOf, Shape of You), which can be seamlessly integrated with user-item interactions. This interaction data is usually presented as a bipartite graph [30]. The crucial point to leverage knowledge graphs to perform item recommendations is to be able to effectively model user-item relatedness from this rich heterogeneous network [31].

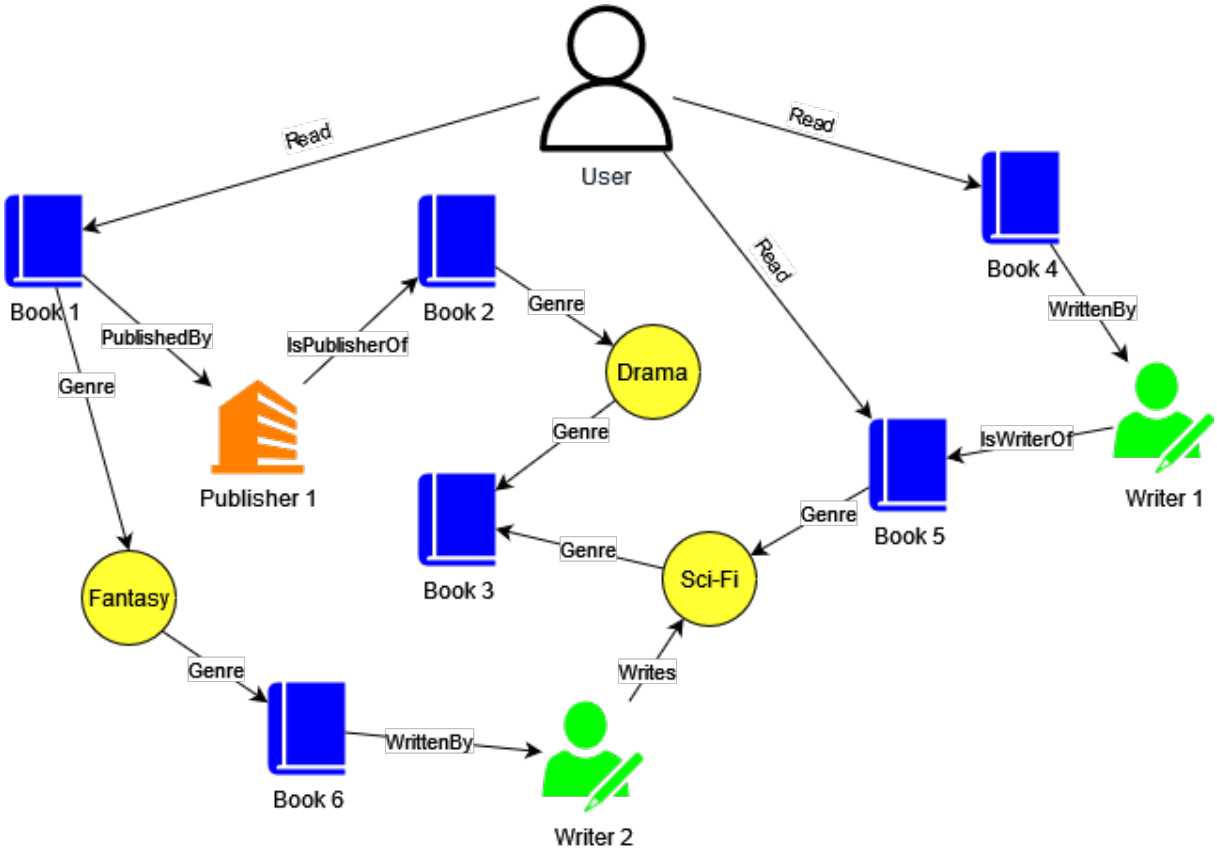

Figure 3: Illustration of KG-aware recommendation

Recommendation methods based on knowledge graphs can be practically categorized into two different types which are Path-based and Embedding-based methods. Where the **Embedding-based** method uses knowledge graph embedding (KGE) techniques trying to learn how to represent users and items. This method uses representation learning to find connections implicitly, rather than using user preferences. Models such as TransE, TransR and TransH build entity and relation embeddings by regarding a relation as translation from head entity to tail entity. These models simply put both entities and relations within the same semantic space. [32]
The **Path-based** method focuses on enhancing the connections called meta-paths that link the users and items, showing how similar they are [33].

2.2.4 Hybrid Approaches

The Hybrid recommender systems try to combine two or more recommendation techniques to get to better performance and accuracy. The most common approach is to have a collaborative filtering technique combined

with some other technique to try to avoid the ramp-up problem. The ramp-up problem actually means two problems, which are "New User" and "New Item" problems (hard to categorize with few ratings).
Different types of hybrid recommender systems exist, here are the following hybdirds listed [15]:

- **Weighted hybrid** - initially gives equal weight to all available recommendation techniques in the system. Gradually adjusts the weighting based on wether the predicted user ratings are confirmed or discomfirmed. The recommended items score is computed from the results.
- **Switching hybrid** - the system switches between recommendation techniques based on some criterion. The advantage is that the system can adapt to the strengths and weaknesses of the different recommendation methods it combines.
- **Mixed hybrid** - recommendations from more than one technique are presented together at the same time.
- **Feature combination** - for example in a content / collaborative merger the system treats the collaborative information as an additional feature data and uses content-based techniques over this enhanced dataset.
- **Cascade hybrid** - one recommender produces a coarse ranking of candidates and than the other refines the recommendations given by the first one. The second step only focuses on the items given by the first step and not all items in the dataset.
- **Feature augmentation** - one technique produces a rating of an item and that information is then incorporated into processing the next recommendation technique. The features used by the second recommender include the output of the first one (like ratings).
- **Meta-level hybrid** - combines two techniques by using the model generated by one as the input for the other, meaning the entire model becomes the input.

2.3 Difficulties related to Recommendation Systems

All types of recommendation systems encounter significant challenges which they have to face and issues they have to solve. Here are some main challenges:

- Cold-start problem - arises when making recommendations to new users and/or items for which the available information is limited. As a result, the recommendations offered in such cases tend to be of poor quality and lack usefulness. [34]
- Data sparsity - when recommender systems use large datasets, the user-item matrix used for filtering can be sparse, which leads to worse performance of recommendations.
- Scalability - as the number of users and items increases, so does the complexity of the algorithms used for recommending items.
- Diversity - helps to discover new products, but some algorithms may accidentally do the opposite, which can also lead to lower accuracy in the recommendation process. [35]
- Privacy - because the information collected by the system usually includes sensitive information that users wish to keep private, users may have a negative impression if the system knows too much about them.
- Serendipity - sometimes can be useful, but if the result of the recommendation system only has serendipitous items and does not have related items, user may think that the system is not reliable. [36]

2.4 Evaluation of Recommendation Systems

When trying to choose which recommendation approach is the best, first it is important to know the use case for the specific system.
The process of finding the most appropriate algorithm for the specific goal typically is based on experiments, comparing the performance of a number of candidate recommenders. Comparing the performance of an algorithm is mostly performed by using some evaluation metric, which usually uses numeric scores, that provides ranking of the compared algorithms.
For measuring the accuracy of predictions of the algorithm three classes of measurements are defined, which are [37]:

- **Measuring Ratings Prediction Accuracy** wishes to measure the accuracy of the system's predicted ratings. The following metrics can be used in such situation: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Normalized RMSE, Normalized MAE, Average RMSE, Average MAE.
- **Measuring Usage Prediction** tries to recommend to users items that they may use, with the following metrics: Precision, Recall (True Positive Rate), False Positive Rate (1 - Specificity), F-measure, Area Under the ROC Curve (AUC).
- **Ranking Measures** are not predicting an explicit rating, but rather are ordering items according to the user's preferences. This can be done Using a Reference Ranking like Normalized Distance based Performance Measure (NDPM), Average Precision (AP) correlation, Spearman's rank correlation coefficient, Kendall's rank correlation coefficient or using Utility-Based Ranking such as Normalized Discounted Cumulative Gain (NDCG), Discounted Cumulative Gain (DCG) or Average Reciprocal Hit Rank (ARHR). Other than that there is also Online Evaluation of Ranking.

The previously mentioned metrics mostly need the feedback from the users, wether it is from ratings or interaction, the metrics are calculated comparing the recommended list with the users data, so knowing which user interacted with which item shows wether the recommendation was accurate or not. However, not all users are trying to use the recommendation engine just for the most accurate predictions, but they might be more interested in other properties of the recommender system like [37]:

- Coverage
- Confidence
- Trust
- Novelty
- Serendipity
- Diversity
- Utility
- Risk
- Robustness
- Privacy
- Adaptability
- Scalability

In the domain of scientific publications, where users are relatively few with respect to the available documents, information needs and interests easily change in an unpredictable way over time due to evolving professional needs, there is also no advertising pushing new items. The Content-based approach does not require particular assumptions over the size and the

activity of the user base. It does not penalize items that have less ratings or are less frequently consumed by many users as long as enough metadata are available, which even allows detailed explanations. These advantages over Collaborative Filtering techniques make this approach particularly attractive to the purpose of providing recommendation in the domain of scientific publications [38].

2.5 Related Works

Several works have explored the application of content-based recommendation systems in the context of digital libraries and book recommendations. Authors in [22] conducted a comprehensive survey on recommender systems in digital libraries, highlighting that more than half of the systems used content-based filtering as their core technique due to the lack of explicit user feedback in academic datasets and the rich textual metadata available for papers and books. Their approach emphasized keyword-based similarity measures, often relying on TF-IDF or simple cosine similarity to recommend research articles.

Authors in [3] presented a structured framework for the evaluation of recommender systems, called the Framework for Evaluating Recommender Systems (FEVR). Their work summarizes insights on how to evaluate recommender systems and organizes them into clear categories, such as evaluation goals, methods, data used, and metrics. The authors argue that all these aspects should be considered together to get meaningful and reliable results. The FEVR framework helps researchers design solid and well-rounded evaluation setups that fit different types of recommender systems.

Authors in [39] surveyed and formalized six key evaluation concepts: utility, novelty, diversity, unexpectedness, serendipity, and coverage. They provided standardized notations and levels of user dependency. Their work critically reflects on how different metrics are connected, categorizing some as user-dependent (e.g., utility, unexpectedness) and others as user-independent (e.g., diversity, coverage). Moreover, they discuss how many metrics overlap in intent and application, and propose that future evaluation frameworks should better align with actual user satisfaction through hybrid online-offline methodologies.

In contrast to these earlier systems, the approach in this work includes Content-based filtering and for similarity measuring using Cosine Similarity applied to different text representation techniques, from traditional TF-IDF and BoW to contextual embeddings like BERT. The evaluation is based on user-independent metrics such as similarity, coverage, diversity, confidence because of the absence of user feedback and user-item interaction.

3 Implementation Proposal

Drawing from the theoretical foundation in the second section, this section proposes a practical framework for evaluating different text representation algorithms tested on recommending books from digital library datasets based on textual metadata and full-text features. We also describe the dataset we will be working with. The proposal includes a detailed explanation of our experiment, along with the rationale behind our choices of specific text represenation methods.

3.1 Dataset Analysis

The main objective of this project is to compare different recommendation algorithms for digital library datasets. To support this, a collection of academic books extracted from the digital library will be used. These books have the text segmented into individual sentences, paragraphs and pages. The experiments will primarily focus on paragraph-level recommendations, as suggesting relevant content based on a single paragraph can be a valuable feature for navigating academic materials in a digital library.

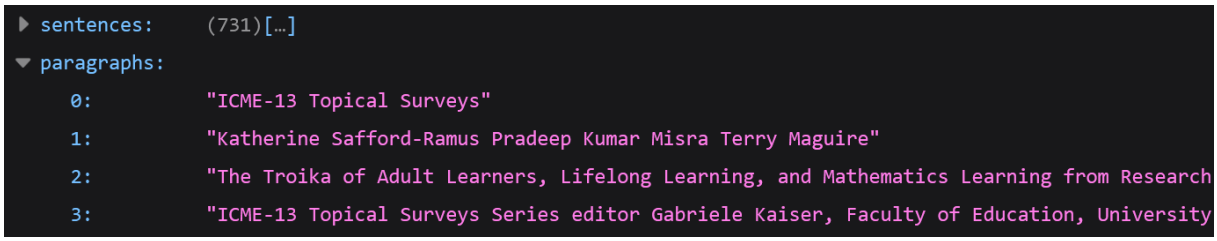

Figure 4: Example of extracted book paragraphs

In addition to this dataset, a second dataset will be used to test the algorithms on simpler and more general types of text. This dataset, from Goodreads, contains book metadata and short descriptions for a wide range of genres including fantasy, comedy, romance, and others. Unlike the academic content of the first dataset, this one includes more casual, narrative-driven books, which helps test the algorithms' performance on short-form and less formal text.

	title	author	rating	no_of_ratings	no_of_reviews	description	genres
0	Divergent	Veronica Roth	4.15	3765886	117,791	In Beatrice Prior's dystopian Chicago world, society i	Young Adult, Dystopia,
1	Catching Fire	Suzanne Collins	4.31	3305054	113,480	Sparks are igniting.Flames are spreading.And the Ca	Young Adult, Dystopia,
2	The Fault in Our Stars	John Green	4.15	4851513	174,662	Despite the tumor-shrinking medical miracle that ha	Young Adult, Romance
3	To Kill a Mockingbird	Harper Lee	4.27	5784553	112,055	The unforgettable novel of a childhood in a sleepy S	Classics, Fiction, Histor
4	The Lightning Thief	Rick Riordan	4.3	2752945	87,446	Alternate cover for this ISBN can be found herePercy	Fantasy, Young Adult, I
5	Harry Potter and the I	J.K. Rowling	4.58	3889833	77,063	Harry Potter, along with his best friends, Ron and He	Fantasy, Fiction, Young
6	The Book Thief	Markus Zusak	4.39	2410045	138,420	Librarian's note: An alternate cover edition can be fo	Historical Fiction, Fictic
7	The Hobbit	J.R.R. Tolkien	4.28	3729821	65,508	In a hole in the ground there lived a hobbit. Not a n	Fantasy, Classics, Fictio
8	Insurgent	Veronica Roth	3.98	1435810	62,356	One choice can transform youâ□□or it can destroy y	Young Adult, Dystopia,
9	Harry Potter and the (	J.K. Rowling	4.43	3669432	73,212	Ever since Harry Potter had come home for the sum	Fantasy, Fiction, Young

Figure 5: Example of book descriptions dataset [2]

By using 2 different datasets in experiments it will ensure that the evaluation covers specialized and general-purpose recommendation use cases. Basic preprocessing such as tokenization, removal of special characters, and optional stopword filtering will be applied prior to embedding.

3.2 Text Representation Methods

In order to build a robust and interpretable content-based recommendation system, a diverse set of seven text representation models is selected:

- TF-IDF
- Bag of Words (BoW)
- Latent Semantic Analysis (LSA)
- GloVe
- FastText
- BERT
- E5

These models chosen represent a broader spectrum of natural language processing techniques, ranging from classical statistical approaches to modern deep learning-based embeddings. This selection has 2 goals. First, to evaluate how different levels of semantic understanding affect recommendation quality and second, to establish clear baselines against which newer models could be compared to later. Each model has its own strengths that help us understand and use text similarity in recommendation tasks more effectively.
Below is an overview of the different methods that will be used for text representation, highlighting their methodology and key characteristics:

1. **TF-IDF (Term Frequency - Inverse Document Frequency)**
   TF-IDF is selected as a well-established and interpretable baseline for measuring textual similarity. It captures term importance based on document frequency and enables fast similarity computation using sparse vectors. TF-IDF creates two scores 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 [5].
2. **BoW (Bag of Words)**
   BoW is probably the most simple method selected here, it provides a valuable lower bound for comparison. It does not consider word order and semantics, making it a good contrast point to evaluate the improvements introduced by more complex models. BoW 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.
3. **LSA (Latent Semantic Analysis)**
   LSA extends TF-IDF by applying dimensionality reduction through Singular Value Decomposition (SVD) to uncover hidden patterns and relationships between terms and documents. This technique helps capture the underlying semantic structure of the text. By decomposing the term-document matrix into three smaller matrices representing topics, terms, and documents LSA identifies latent topics and represents documents in a lower-dimensional semantic space. [40].
4. **GloVe (Global Vectors for Word Representation)**
   GloVe is included as a pretrained static word embedding model that represents words as fixed-size vectors based on how often they appear

together in a large text corpus. It 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 to learn the relationships between words, allowing it to capture their meanings and similarities. GloVe helps to test how well combining individual word vectors can represent the meaning of a full document.

5. **FastText**
   FastText enhances traditional word embeddings by incorporating subword information, allowing the model to generate embeddings even for rare or complex words. 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 [8].

6. **BERT (Bidirectional Encoder Representations from Transformers - SentenceTransformer)**
   BERT represents the state-of-the-art in contextual language modeling. Unlike the other models, BERT produces embeddings that consider the full sentence context, allowing it to distinguish between different meanings of the same word depending on usage. It uses attention mechanisms to model relationships between all words in a sentence allowing bidirectional encoding. [41]

7. **E5 (Embedding-based Encoder for Information Retrieval)**
   E5 is a transformer-based model, designed specifically for dense retrieval and semantic similarity tasks. Similar to BERT, it produces contextual embeddings at the sentence level, capturing the underlying meaning of text rather than relying on surface-level token matches. E5 excels in matching short texts such as questions, passages, or document segments based on semantic relevance.

By including models from these various categories (statistical, latent, word-level embeddings, and contextual transformer-based representations) the experimental design ensures a wider comparison across different levels of linguistic representation.

3.3 Experiment Types

Experimenting with recommendations can be done in different ways. Here are the types of experiments to test the algorithms:

Type	Description
Offline	Method: simulation of user behavior based on past interactions
	Task: defined by the researcher, purely algorithmic
	Repeatability: evaluation of an arbitrary number of experiments possible at low cost
	Scale: large dataset, large number of users
	Insights: quantitative, narrow
User Study	Method: user observation in live or laboratory setting
	Task: defined by the researcher, carried out by the user
	Repeatability: expensive
	Scale: small cohort of users
	Insights: quantitative and/or qualitative
Online	Method: real-world user observation, online field experiment
	Task: self-selected by the user, carried out by the user
	Repeatability: expensive
	Scale: large
	Insights: quantitative and/or qualitative

Table 1: Overview of Experiment Types [3]

Offline experiments are the most popular experiment type. They aim to compare different recommendation algorithms and settings and they do not require any user interaction and may be considered system-centric [3]. The **offline evaluation** method will be adopted, as the goal is to test and compare recommendation algorithms without relying on actual user interaction or feedback. This provides a practical way to test the system without users and could be repeated as many times as needed for system-centric experimentation.

3.4 Conceptual Proposal

1. **Loading raw data from JSON files**
   - Paragraphs will be extracted from academic books into a structured format, with book and paragraph index

2. **Including an additional dataset of book descriptions**
    - A second dataset containing book metadata and textual descriptions (from Goodreads) will be used
    - This dataset includes books from various genres such as fantasy, comedy, and romance, with the goal of testing on short-form, non-academic content
3. **Preprocessing and organization**
    - Both datasets will be organized in DataFrames for consistent handling
    - Text will be cleaned and normalized to prepare it for embedding
4. **Implementation of the recommendation algorithm**
    - A pipeline will be developed for generating recommendations based on paragraph-to-paragraph and description-to-description similarity
    - The system will support multiple text representation techniques
5. **Integration of different text representation methods**
    - TF-IDF (Term Frequency–Inverse Document Frequency)
    - LSA (Latent Semantic Analysis)
    - BoW (Bag of Words)
    - FastText (Facebook's FastText Word Embeddings)
    - GloVe (Global Vectors for Word Representation)
    - BERT (Bidirectional Encoder Representations from SentenceTransformers)
    - E5 (Embedding-based Encoder for Information Retrieval from SentenceTransformers)
6. **Running experiments and storing results**
    - Each model will be executed independently on both datasets
    - Top-N recommendations will be generated for selected input texts
7. **Evaluation of recommendations**
    - Evaluation will be based on metrics

- Time and memory usage will be tracked to compare algorithm performance

4 Implementation

This section describes how the proposed system was implemented in practice. It outlines the data preparation process, integration of multiple text representation methods, the overall recommendation pipeline and the development environment.

The system was implemented using the **Python** programming language, which is widely adopted in the fields of data science and machine learning. For data preparation and experimentation, the **Jupyter Notebook** environment was used. This way the development was interactive with step-by-step testing of individual components during preprocessing and exploration of various recommendation models.

While the notebooks were used locally for developing and testing the recommendation logic, the final recommendation system was implemented as a collection of structured Python scripts. These scripts were executed on a **Google Cloud Virtual Machine (VM)** configured with 8 vCPUs, 32 GB of RAM, and a dedicated NVIDIA L4 GPU. This cloud-based deployment enabled efficient GPU-accelerated execution of models and ensured clean, uninterrupted **multiprocessing-based resource tracking** across CPU, memory, and GPU usage.

Each model was tested in ten evaluation runs using this setup, with results logged for further evaluation. Then the evaluation itself was carried out in a separate Jupyter Notebook, executed locally, where the recommendation results from the VM runs were collected and visualized.

4.1 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. These JSON files were loaded and parsed into *Pandas DataFrames* for easier handling and segmentation into paragraphs. The processed data was then exported into CSV files and a dataset of 45,461 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 [2] was

also used in the experiments. This dataset includes metadata for over 13,000 books, such as title, author, genres, ratings, and textual descriptions. The dataset contains entries with descriptions averaging approximately 160 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 comparison.

4.2 Content-Based Recommendation Pipeline

The core of the system is a recommendation pipeline that is able to use various text-based content recommendation models. Each model inherits from an abstract base class, ensuring a unified interface for training, input preparation, similarity computation, and output formatting.

The recommendation process begins by loading a configuration object that defines the algorithm to be used, the dataset path, the thresholds and the target input samples (book or paragraph). Once the appropriate model is initialized, the following steps are executed:

1. **Model Training:**
   The selected model is trained on the dataset using its internal representation logic. Some models (e.g., TF-IDF, FastText) require training on the dataset, while others (e.g., GloVe, BERT) use pre-trained embeddings and only load them without fitting on the current data [42].
2. **Input Preparation:**
   The input book description or paragraph is extracted based on predefined pairs of indices from the configuration. A filtered dataset is prepared by excluding the current input from the dataset to prevent self-recommendation.
3. **Vectorization or Embedding:**
   The input text and all other documents from the filtered set are converted into fixed-size vector representations. This process depends on the model, using bag-of-words, TF-IDF weights, or dense embeddings (e.g., GloVe, FastText, BERT).
4. **Similarity Calculation:**
   Cosine similarity is used to compare the input vector with each document vector. If the model uses matrix operations (e.g., BoW, TF-IDF), the similarities are computed in batch for improved performance. For

embedding-based models, similarity is computed individually for each document embedding.

5. **Filtering and Ranking:**
   Results are filtered using a dynamic threshold that is also loaded from the configuration, calculated as a percentage of the maximum similarity score, ensuring that only highly similar documents are returned. The top N documents are then selected based on their similarity values, or if the coverage is to be tested, then all the recommendations are returned above the threshold.
6. **Result Storage and Tracking:**
   For each execution, the input and the list of recommended books (with metadata and similarity score) are stored in CSV files. Additionally, resource usage (CPU, GPU, memory) and execution time are logged for performance evaluation.

The pipeline is flexible, allowing new models to be integrated with minimal configuration. Each model defines its own internal logic but uses the same recommendation pipeline and tracking interface to keep fair and consistent benchmarking across multiple text representation approaches.

4.3 Deployment

The following diagram presents the components of the system showing the communication between parts of the system:

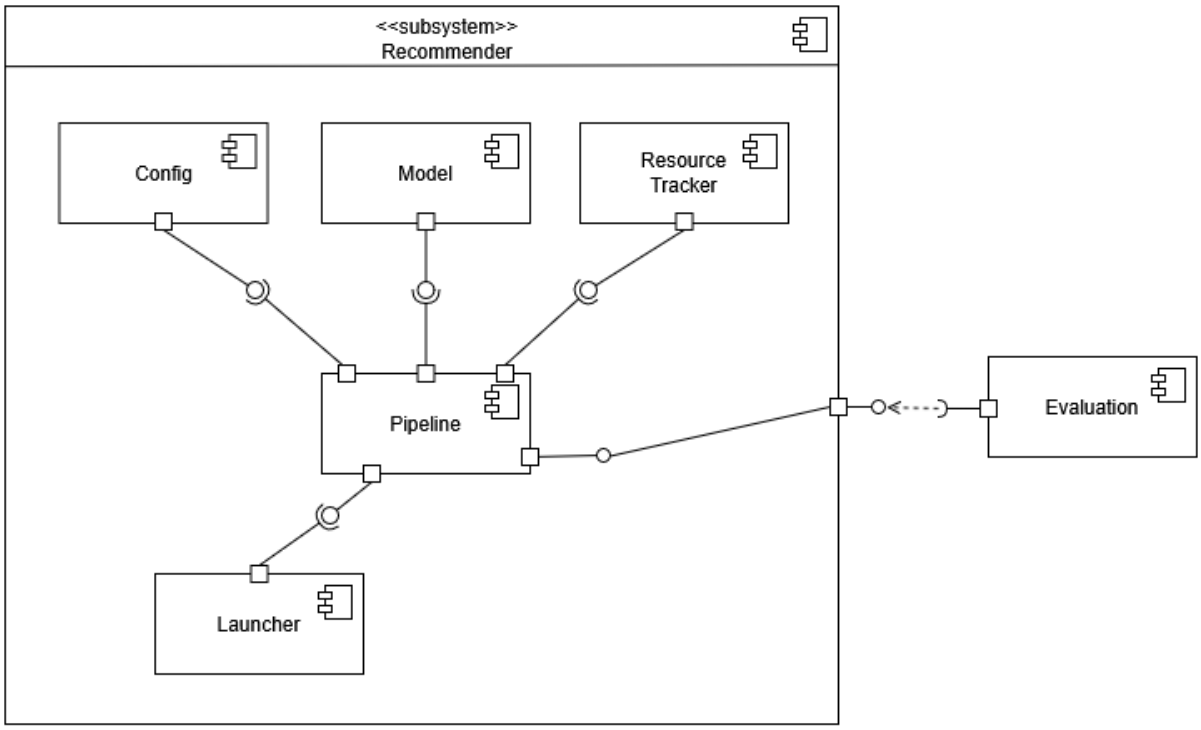

Figure 6: Recommender system component diagram

Here is the deployment setup of the system, including the virtual machine environment and local tools used for development and evaluation:

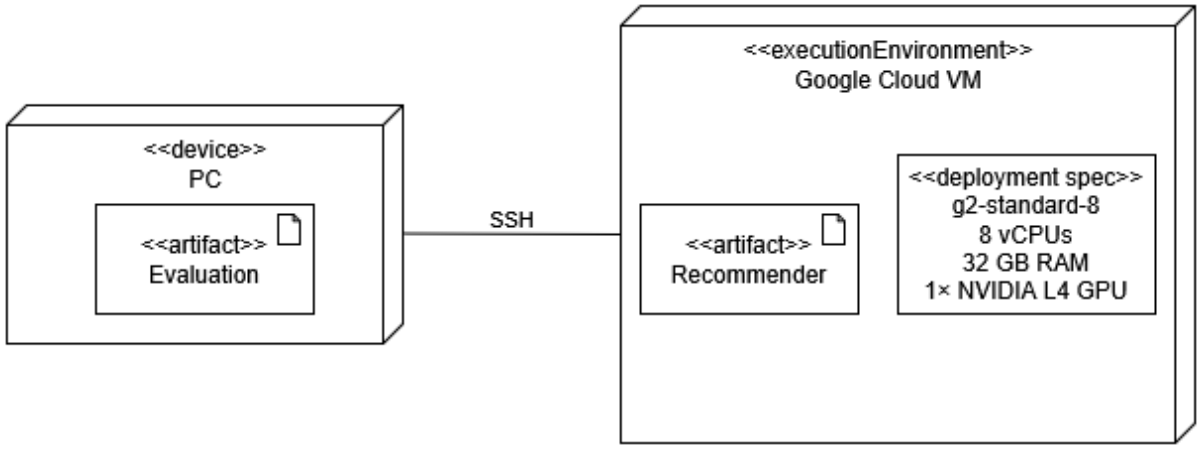

Figure 7: Deployment diagram

5 Evaluation and Results

This section presents how the recommendation results were evaluated. The goal is to compare the previously mentioned text representation methods, and for comparison there are different metrics to prove which algorithms perform better in which scenarios. When **Evaluating** a RS there are two main types of evaluation [3], which are:

- System-Centric Evaluation
  - Algorithmic Aspects - e.g., the predictive accuracy of recommendation algorithms
- User-Centric Evaluation
  - Users' Perspective - how users perceive its quality or the user experience when interacting with the RS

We will focus on System-Centric Evaluation to test the performance of the algorithms and show results based on the metrics listed below.

5.1 Metrics for Evaluation

It is crucial to define how Relevant an item is based on the input item. Since the digital library dataset used in this work does not track user ratings, interactions, or feedback yet, traditional evaluation metrics that rely on relevance from interaction such as Precision, Recall, or NDCG cannot be applied. The dataset provided the full text of books segmented into paragraphs, without any indication of which content users found useful or interesting. So evaluation had to be performed in a user-independent setting based on the internal characteristics of the recommendation results, which meant it was calculated from content similarity between items. The metrics below describe how the different text representational models were compared, were the Coverage was calculated from a dynamic threshold and the other metrics were computed using the top 10 recommendations per input. This choice was made to focuses on the most similar outputs of each algorithm.

- **Similarity** – captures how similar each recommended item is to the input item. It is computed using cosine similarity between vectorized representations of the text. The average similarity score across rec-

ommendations is used to evaluate how closely the algorithm matches content. Higher similarity will show higher relevance between the compared items in the list of recommended items.

- **Diversity** – measures how different the recommended items are from each other, helping to avoid redundant suggestions and offer a wider range of content. It was calculated using two approaches:
  - *Variance of Similarity Scores:* This method checks how spread out the similarity scores are among recommended items. Higher variance means the items are more varied.
  - *Average Dissimilarity:* This calculates diversity as 1−cosine similarity, then averages these values to show how distinct the items are from one another.

  Both methods help ensure that the recommendation list includes a mix of different content types. High diversity improves the chance of covering more topics. These approaches follow the method described Authors in [39].

- **Confidence** – shows how certain the system is about its recommendations, which was also sampled from top 10 recommendations. It is based on the similarity scores between the input item and the recommended items. Confidence is calculated as the inverse of the standard deviation of these similarity scores:

$$\text{Confidence} = \frac{1}{1 + \text{std\_dev}(\text{similarities})}$$

  This results in a value between 0 and 1. A higher confidence means that the recommendations are not only strongly related to the input but also consistently similar to each other, indicating that the system is more sure about its recommendations.

- **Coverage** – Catalog Coverage evaluates how effectively a recommendation algorithm utilizes the dataset, measured as the proportion of items in the dataset that are recommended. High item-space coverage indicates that the algorithm explores a wide range of items and is not limited to a small subset. It is calculated as:

$$\text{Coverage} = \frac{\text{Number of unique recommended items}}{\text{Total number of items in the dataset}} \times 100$$

To balance the number and relevance (similarity) of recommendations, we used a dynamic thresholding approach so only items scoring above the threshold were recommended. It was set up as a proportion of the maximum similarity per input. Sparse vector models TF-IDF, BoW, LSA and dense BERT embeddings produced lower similarity scores, so a lower threshold (0.5) was used. On the other hand FastText, E5, GloVe returned high similarity scores even for weak matches, requiring stricter thresholds (over 0.9) to avoid overly broad results. We chose these threshold settings to balance recommendation similarity with dataset coverage.

In our setup the dataset from GoodReads books contained metadata information to the items such as ratings, number of ratings, number of review and genres. Although this dataset did not include interaction histories (which user selected which book), it allowed for a content-level analysis of recommendation quality through this additional information.

- **Novelty** - refers to the fraction of recommended items new to the user, which means they are less known or uncommon items. This metric is based on the assumption that highly rated and popular items are likely to be known to users and therefore not novel. A good measure for novelty might be to look more generally at how well a recommendation system made the user aware of previously unknown items that subsequently turn out to be useful in context. [43]
  For our results items were labeled as *novel* if they scored below all three novelty thresholds: below the 25th percentile for rating, number of ratings, and number of reviews. The novelty score was computed as the percentage of top-N recommendations falling into this category.
- **Popularity** – captures how frequently recommended items are known or mainstream. Items were labeled as *popular* if they exceeded all three of the following thresholds, derived from the full dataset: above-average rating, above the 75th percentile in number of ratings, and above the 75th percentile in number of reviews. The popularity score of a recommendation set was then calculated as the proportion of top-N recommended items classified as popular.

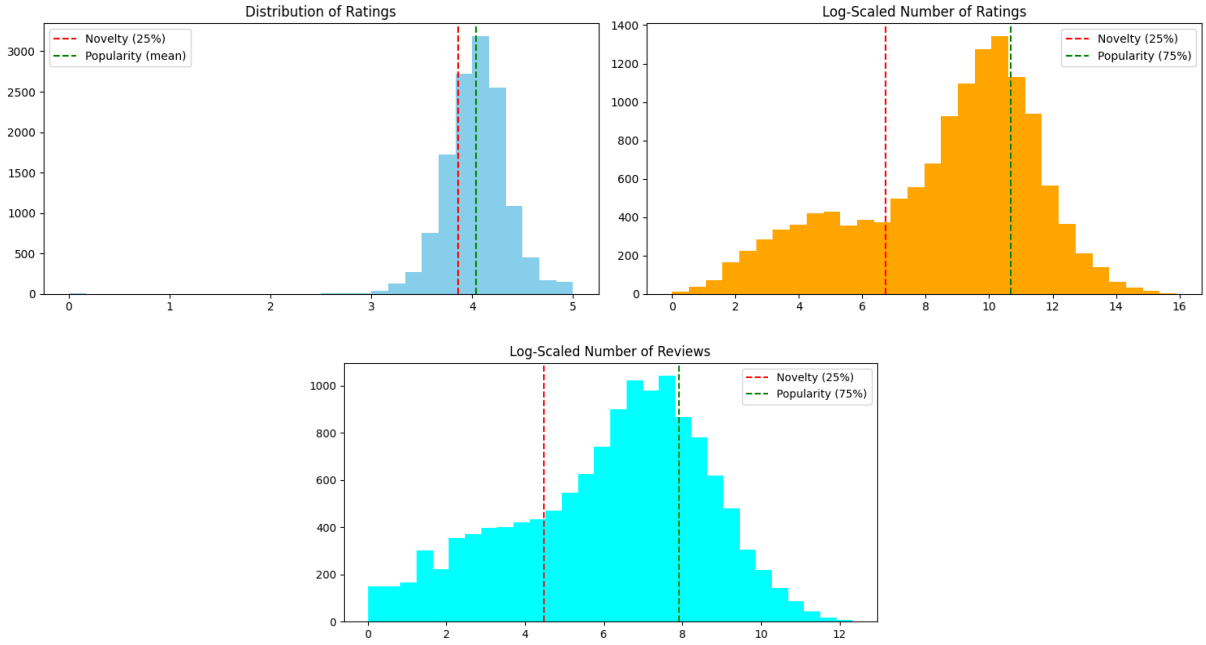

Figure 8: Distribution of ratings and reviews in GoodReads dataset

5.2 Results

The radar charts in Figures 9 and 10 present a visual comparison of each algorithm's performance across different evaluation metrics. All values are normalized to a $[0, 1]$ scale to be able to compare them across metrics with different units. The metrics include similarity, confidence, coverage, diversity variance and diversity dissimilarity measures. For the book-description dataset, additional axes for popularity and novelty ratio are also included. Figure 9 shows that GloVe, E5 and Fasttext achieved the highest scores in both coverage and confidence, highlighting their ability to recommend a wide range of items with high semantic consistency. BERT produced the most diverse recommendations in terms of similarity variance, while TF-IDF and BoW scored highest in dissimilarity-based diversity, which means that they returned more varied and less semantically redundant items. Interestingly, BoW also achieved the top scores for popularity and along with E5 they had the highest score for novelty. This may be due to the way BoW works, since it relies on simple keyword overlap. It can recommend very popular items if the input shares common words with them, but it can also suggest less-known books that match on rare or unique words. Because of this, BoW tends to give either very well-known or very unusual results, which helps explain its high scores in both popularity and nov-

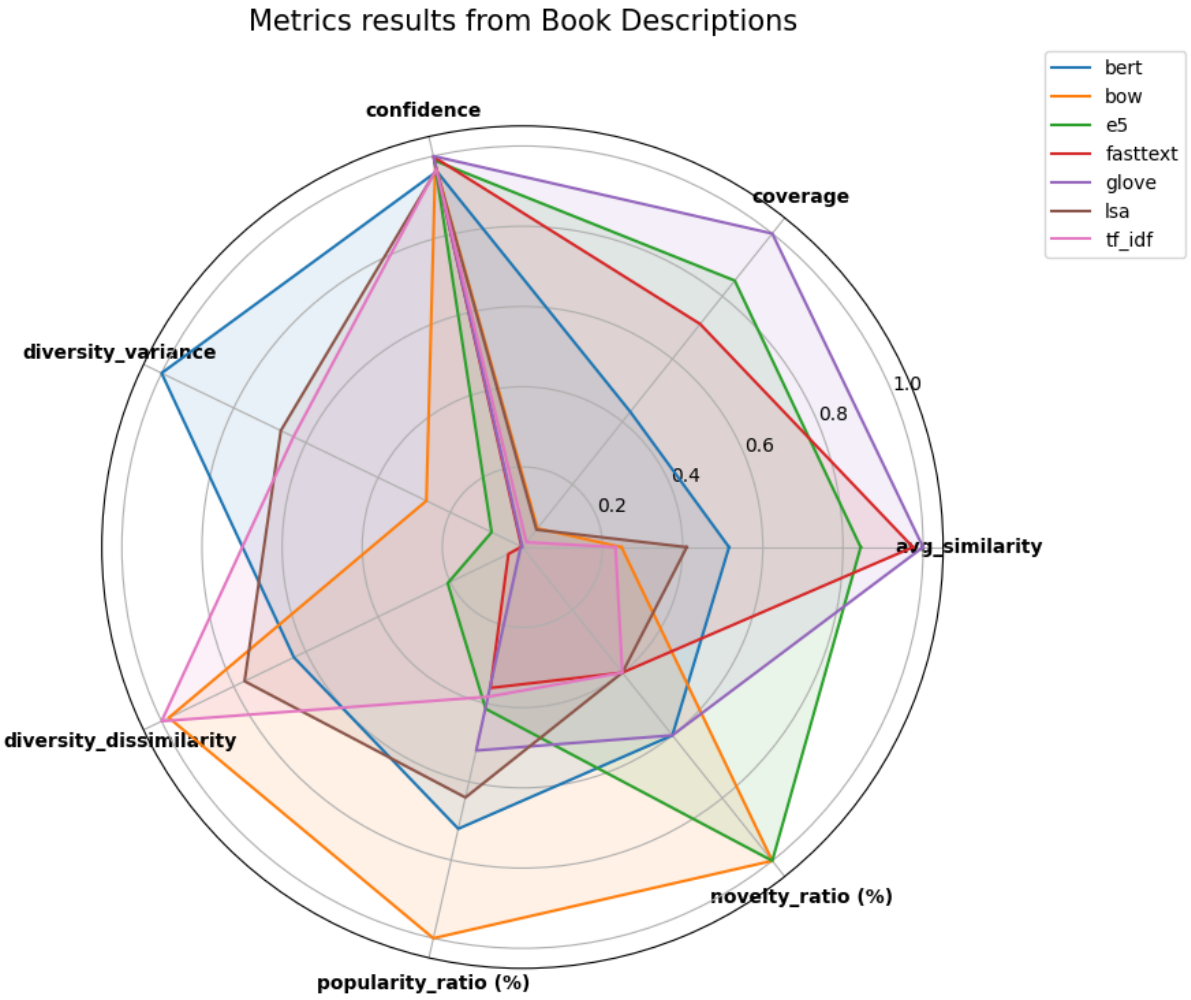

Figure 9: Metrics Results of Descriptions Vizualized

elty. E5 embeds entire sentences into dense semantic vectors, capturing deeper contextual meaning, which helps it recommend less typical but still relevant books, which could have resulted in high novelty.

Model	Sim	Cov	Conf	Div (Var.)	Div (Diss.)	Pop (%)	Nov (%)
bert	0.513	9.29	0.96039	0.0022251719	0.4866	18.00	3.00
bow	0.246	1.29	0.97755	0.0005924999	0.7530	25.00	5.00
e5	0.84	18.29	0.9883	0.0001904935	0.1593	10.33	5.00
fasttext	0.970	15.31	0.99765	0.0000059949	0.0299	9.00	2.00
glove	0.995	21.52	0.99946	0.0000002640	0.0045	13.00	3.00
lsa	0.408	1.20	0.96473	0.0014899120	0.5917	16.00	2.00
tf_idf	0.231	0.34	0.96648	0.0014116538	0.7684	9.56	2.00

Table 2: Results of Metrics for book descriptions

Algorithm	CPU Time (s)	RAM (MB)	GPU Mem (MB)
BERT	27.757	1141.38	563.78
E5	106.777	1200.35	1732.01
BoW	0.275	202.89	
FastText	11.821	1193.43	
GloVe	3.210	344.33	
LSA	2.041	424.85	
TF-IDF	1.729	193.64	

Table 3: Average resource usage for recommending Book Descriptions

Algorithm	CPU Time (s)	RAM (MB)	GPU Mem (MB)
BERT	103.399	1227.44	670.89
E5	482.157	1313.28	1824.2
BoW	0.809	378.45	
FastText	44.964	2002.80	
GloVe	10.314	420.78	
LSA	8.179	744.64	
TF-IDF	7.230	293.49	

Table 4: Average resource usage for recommending Book Paragraphs

The resource usage of each algorithm is shown in Tables 3 and 4. Deep learning-based models such as BERT, E5 and FastText consumed significantly more CPU time and memory (RAM/GPU), reflecting the cost of their improved similarity results. Traditional models like BoW, TF-IDF, and LSA were computationally lightweight and offered faster execution, which may be better or some systems would prefer it in low-resource environments.

Figure 10 presents the evaluation results based on paragraph-level recommendations. FastText and GloVe achieved the highest scores in similarity, highlighting their effectiveness in capturing semantic closeness between input and recommended items. E5 and FastText showed the strongest performance in coverage, indicating their ability to recommend a wide range of items across the dataset. Confidence values were highest for FastText, E5 and GloVe, although all models had relatively similar scores, suggesting consistent recommendation strength.

In contrast, BoW achieved the highest diversity in variance and TF-IDF achieved it in dissimilarity measures, reflecting their ability to produce a wider range of less semantically redundant results. This comes at the cost of lower similarity, indicating a broader but potentially less targeted recommendation list. BERT provided a balanced performance across most metrics, offering moderately strong similarity, confidence, and diversity.

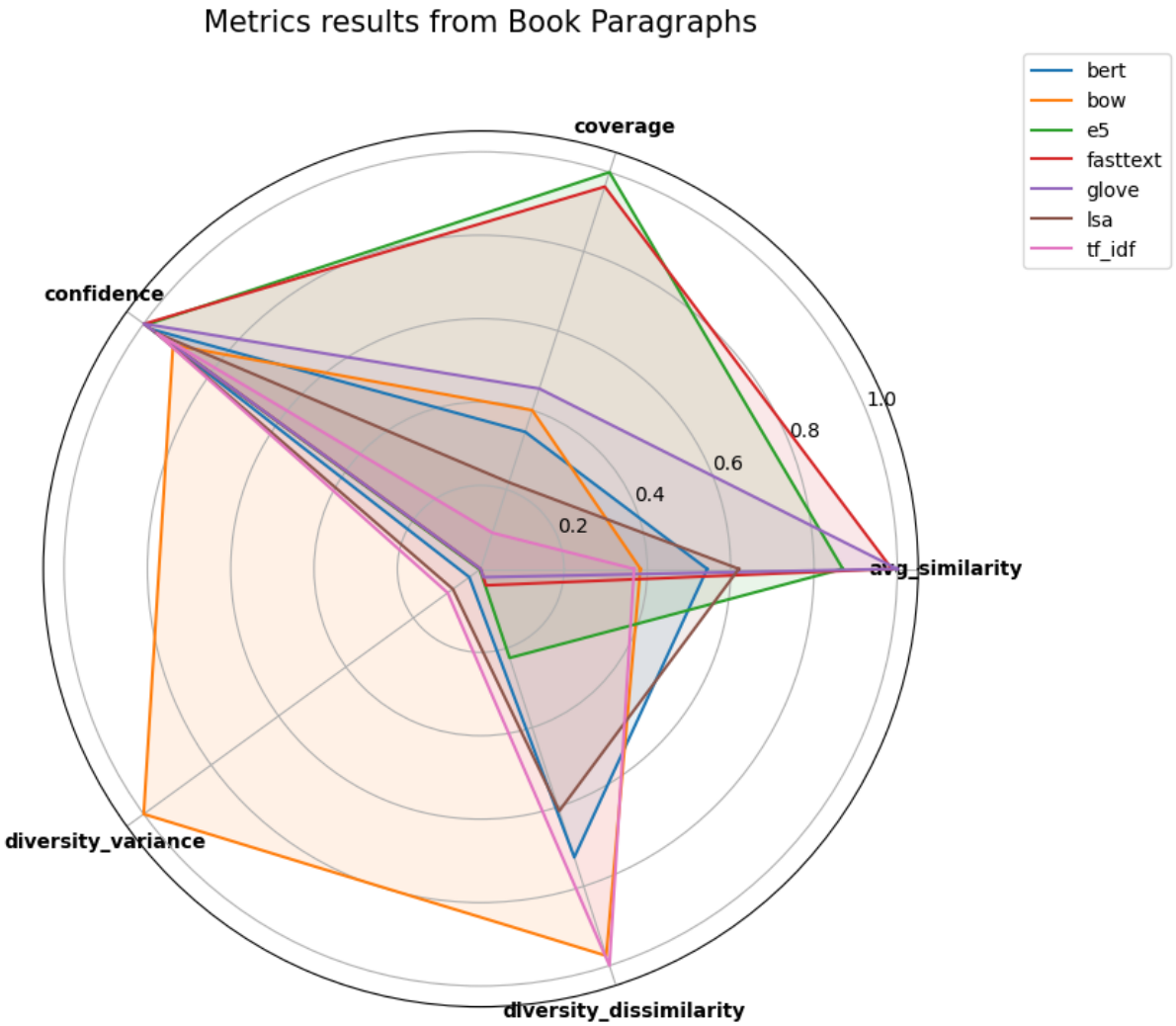

Figure 10: Metrics Results of Paragraphs Vizualized

Model	Sim	Cov	Conf	Div (Var.)	Div (Diss.)
bert	0.536	0.19	0.97654	0.0006018049	0.463
bow	0.378	0.22	0.91190	0.0178139560	0.621
e5	0.857	0.55	0.99352	0.000048741	0.1429
fasttext	0.973	0.53	0.99791	0.0000068039	0.0264
glove	0.986	0.25	0.99877	0.0000019924	0.0134
lsa	0.611	0.12	0.96231	0.0014689289	0.3886
tf_idf	0.363	0.05	0.95945	0.0017445470	0.6368

Table 5: Results of Metrics for book paragraphs

6 Conclusion

The field of recommendation algorithms is very popular in academic research. In order to know which type of recommendation method to use our work was set out to evaluate the effectiveness of various content-based recommendation algorithms when applied to digital library datasets. For comparison two distinct datasets were used in this work, one based on academic book paragraphs and the other from Goodreads containing book descriptions and associated metadata. The project explored the strengths and limitations of seven different text representation methods: *TF-IDF*, *BoW*, *LSA*, *GloVe*, *FastText*, *BERT* and **E5**, while recommending items based on *Cosine Similarity*. To compare the models in a fair and consistent manner we used a system-centric evaluation approach and we defined metrics such as similarity, diversity, confidence, coverage, novelty, and popularity. These were selected to reflect how similar, consistent, and varied the recommendations were, especially in the absence of user interaction data. Besides those metrics resource consumption was tracked, which showed insights of each algorithm's performance. To conclude, the results demonstrated that dense embedding models like GloVe, E5 and FastText excelled at finding semantically similar items with strong coverage and confidence, while traditional models like BoW and TF-IDF offered higher diversity and showed some unexpected results in achieving high popularity and novelty scores. BERT and E5 were more resource-intensive, but BERT offered a well-balanced performance across most metrics. The evaluation pipeline and multi-metric approach ensure that the findings are reproducible and can be adapted for other datasets or domains in the future.

6.1 Future Work

While the current system benchmarks several key algorithms in a content-based setting for textual recommendations, there are several directions to further develop and expand the framework. Adding more state-of-the-art models, especially recent advancements in language representation,could make the recommendations even more accurate. The system could also be tested on more types of datasets, like multilingual content, other forms of content or datasets from different domains. There could also be imrpovements made in cleaning and preprocessing the text better as in normalizing

or tokenizing differently. Right now, the system only uses content-based methods. In the future, it could also include collaborative or hybrid recommendation approaches, which combine different techniques. However, implementing these approaches would require user interaction data through clicks or ratings to be able to define which items the user selected and better define which recommendations are more relevant.

Resumé

Úvod

Odporúčacie systémy sú čoraz dôležitejšie pri filtrovaní veľkého množstva online informácií. Nejde len o úsporu času, ale aj o zlepšenie používateľského zážitku tým, že systém predpovedá, o aký obsah by mohol mať používateľ záujem. Tento projekt sa zameriava na odporúčanie kníh pomocou rôznych algoritmov a ich vyhodnotenie pomocou stanovených metrík.

Pochopenie odporúčacích systémov

Výskum sa zaoberá porozumením odporúčacích systémov, ich základnými princípmi, typmi a technikami, ktoré sa používajú na generovanie personalizovaných odporúčaní. Predstavili sme rozdiel medzi odporúčacími systémami a vyhľadávačmi. **Odporúčacie systémy** navrhujú personalizovaný obsah na základe predchádzajúceho správania a preferencií používateľa, zatiaľ čo **vyhľadávače** vyhľadávajú informácie na základe zadaných kľúčových slov.

Popísali sme hlavné typy odporúčaní vrátane odporúčaní založených na obsahu (content-based), kolaboratívneho filtrovania (collaborative filtering), znalostných grafov (knowledge graphs) a hybridných prístupov. Zamerali sme sa na ich výhody, nevýhody, architektúru a metódy ako napríklad maticovú faktorizáciu, využitie sémantických informácií a aj akým spôsobom používajú spätnú väzbu, či už explicitnú (napr. hodnotenia, komentáre od používateľa) alebo implicitnú (napr. počet kliknutí, čas strávený na stránke, história prezerania).

V časti o ťažkostiach odporúčacích systémov sme poukázali na problémy ako studený-štart, riedkosť dát, škálovateľnosť, súkromie a serendipitu. Dôležitou súčasťou analýzy bola aj časť o hodnotení týchto systémov, kde sme rozdelili metriky na tie zamerané na presnosť predikcie, predikciu použitia a poradie. Okrem toho sme spomenuli ďalšie metriky ako napríklad dôvera, rôznorodosť, pokrytie, robustnosť a sebaistota odporúčania.

V rámci prehľadu prác v oblasti odporúčacích systémov Beel et al. (2016) ukázali, že viac ako polovica odporúčacích systémov v akademickom prostredí používa odporúčanie založené na obsahu, najmä kvôli absencii spätnej

väzby a bohatého textu [22].

Návrh riešenia

Na základe analýzy bola navrhnutá architektúra systému, ktorý porovnáva viacero algoritmov na odporúčanie kníh. Boli použité dva datasety – prvý obsahuje akademické knihy rozdelené na odseky a druhý obsahuje opisné metadáta z knižnej platformy Goodreads, pokrývajúce rôzne žánre ako fantasy, komédia či romantika. Prístup umožňuje otestovať odporúčania v rôznych kontextoch a textových štruktúrach.

Navrhnutý systém využíva viacero techník na reprezentáciu textu vrátane:

- TF-IDF – Zohľadňuje dôležitosť slov podľa ich výskytu v dokumente a v celom korpuse; vytvára riedke vektory s váhami slov.
- Bag of Words (BoW) – Jednoduchá metóda založená na počte výskytov slov bez ohľadu na poradie alebo význam; každé slovo je reprezentované ako samostatná dimenzia.
- Latent Semantic Analysis (LSA) – Rozširuje TF-IDF pomocou zníženia dimenzionality (SVD), čím odhaľuje skryté vzťahy medzi slovami a dokumentmi.
- GloVe – Predtrénovaný model vektorov slov založený na frekvencii spoločného výskytu slov v širokom kontexte; vhodný na zachytenie významových vzťahov.
- FastText – Vylepšuje vektorové reprezentácie slov pomocou znakových n-gramov, čo umožňuje reprezentovať aj neznáme alebo zriedkavé slová.
- BERT – Kontextuálny jazykový model využívajúci mechanizmus vlastnej pozornosti (self-attention), ktorý vytvára vektory na úrovni viet s ohľadom na význam slov v danom kontexte.
- E5 –

Tieto metódy pokrývajú spektrum od jednoduchých štatistických modelov až po pokročilé vektorové reprezentácie učené pomocou hlbokých neurónových sietí.

Implementácia

Systém bol implementovaný v jazyku Python. Použitím knižníc ako pandas, scikit-learn a transformers bolo možné spracovať a implementovať modely na reprezentáciu textu. Kvôli výpočtovo náročním modelom bola použitá virtuálna mašina v Google Cloud s GPU podporou, kde boli jednotlivé algoritmy spustené. Na ukladanie výsledkov odporúčaní, spotreby zdrojov a meraní bol vytvorený jednotný postup. Základný postup implementácie obsahoval tieto kroky:

- Načítanie a predspracovanie dát
- Vytvorenie modelov na reprezentáciu textu
- Výpočet podobnosti medzi položkami pomocou kosínusovej podobnosti
- Filtrovanie a zoradenie výsledkov
- Meranie výkonnosti a vyhodnotenie metrických výsledkov

Vyhodnotenie a výsledky

Všetky modely boli porovnané pomocou viacerých metrík:

- Podobnosť (Similarity) – miera podobnosti medzi odporúčanými a vstupnými textami
- Dôvera (Confidence) – stabilita odporúčaní, založená na podobnostiach medzi vstupom a odporúčanými položkami; počítaná ako inverzná hodnota smerodajnej odchýlky týchto podobností, kde vyššia hodnota znamená konzistentnejšie odporúčania.
- Rôznorodosť (Diversity) – rôznorodosť odporúčaných položiek, meraná dvoma spôsobmi: (1) ako rozptyl skóre podobnosti medzi odporúčaniami (vyšší rozptyl znamená väčšiu rôznorodosť) a (2) ako priemerná nepodobnosť, počítaná ako opak podobnosti.
- Pokrytie (Coverage) – podiel odporúčaných položiek z celkového datasetu
- Novinka (Novelty) – výskyt menej známych alebo hodnotených položiek
- Popularita (Popularity) – podiel často hodnotených alebo populárnych položiek

Na základe metrík a analýzy výkonu boli pozorované nasledovné výsledky:

- FastText a GloVe dosiahli vysokú podobnosť, pokrytie a dôveru.
- BERT priniesol najvyváženejšie výsledky, ale s vyššou náročnosťou na výpočtové zdroje.
- Tradičné modely ako BoW a TF-IDF ponúkli vysokú diverzitu a zároveň zaujímavý pomer populárnych aj nových odporúčaní.

Okrem kvalitatívneho hodnotenia boli zaznamenané aj časy spracovania, spotreba RAM a GPU pamäte. Tieto informácie umožňujú lepšie rozhodovanie o výbere modelov v rôznych prostrediach.

Záver

Práca preukázala, že výber reprezentácie textu výrazne ovplyvňuje výsledky odporúčania. Moderné metódy ako FastText či GloVe poskytujú vysoko podobné odporúčania, zatiaľ čo jednoduchšie metódy ako BoW môžu priniesť väčšiu rôznorodosť a popularitu. Navrhnutý systém je modulárny a rozšíriteľný, čo umožňuje jeho ďalší vývoj v akademickom aj praktickom prostredí.

Možnosti ďalšieho rozšírenia

Budúci vývoj systému by mohol zahŕňať: zapojenie novších modelov na spracovanie prirodzeného jazyka (napr. veľké jazykové modely), testovanie na viac typoch datasetov (viacjazyčné texty), vylepšenie predspracovania textu a čistenia údajov, začlenenie kolaboratívneho alebo hybridného odporúčania, ktoré si však vyžaduje dáta so spätnou väzbou od používateľov. V konečnom dôsledku je systém vhodný na ďalšie testovanie, rozšírenie do webovej aplikácie alebo reálne použitie v digitálnych knižniciach.

References

[1] Cataldo Musto, Marco de Gemmis, Pasquale Lops, Fedelucio Narducci, and Giovanni Semeraro. *Semantics and Content-Based Recommendations*. 2022. doi:10.1007/978-1-0716-2197-4_7.

[2] Deepak Kumar. Goodreads book data. `https://www.kaggle.com/datasets/deepaktheanalyst/books-details-dataset`, 2022. Accessed: April 12, 2025.

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

[4] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to Recommender Systems Handbook*, pages 1–35. 2010. doi:10.1007/978-0-387-85820-3_1.

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

[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] Serge Stephane AMAN, Behou Gerard N'GUESSAN, Djama Djoman Alfred AGBO, and KONE Tiemoman. Search engine performance optimization: methods and techniques. 12, 2024. doi:10.12688/f1000research.140393.3.

[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] 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.

[12] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Content-based Recommender Systems: State of the Art and Trends*, pages 73–105. 2010. doi:10.1007/978-0-387-85820-3_3.

[13] Mehrbakhsh Nilashi, Othman Ibrahim, and Karamollah Bagherifard. A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. 92:507 – 520, 2018. doi:10.1016/j.eswa.2017.09.058.

[14] 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.

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

[16] 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.

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

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

[19] X. Ning, C. Desrosiers, and G. Karypis. *A comprehensive survey of neighborhood-based recommendation methods*. 2015. doi:10.1007/978-1-4899-7637-6_2.

[20] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. 42(8):30 – 37, 2009. doi:10.1109/MC.2009.263.

[21] Srilatha Tokala, Murali Krishna Enduri, T. Jaya Lakshmi, and Hemlata Sharma. Community-based matrix factorization (cbmf) approach for enhancing quality of recommendations. 25(9), 2023. doi:10.3390/e25091360.

[22] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. Research-paper recommender systems: a literature survey. 17(4):305 – 338, 2016. doi:10.1007/s00799-015-0156-0.

[23] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. page 195 – 204, 2000. doi:10.1145/336597.336662.

[24] Pasquale Lops, Dietmar Jannach, Cataldo Musto, Toine Bogers, and Marijn Koolen. Trends in content-based recommendation: Preface to the special issue on recommender systems based on rich item descriptions. 29(2):239 – 249, 2019. doi:10.1007/s11257-019-09231-w.

[25] M. De Gemmis, P. Lops, C. Musto, F. Narducci, and G. Semeraro. *Semantics-aware content-based recommender systems*. 2015. doi:10.1007/978-1-4899-7637-6_4.

[26] Shilpa S. Laddha and Pradip M. Jawandhiya. Semantic search engine. 10(21):1–6, 2017. doi:10.17485/ijst/2017/v10i23/115568.

[27] Ran Huang. Improved content recommendation algorithm integrating semantic information. 10(1), 2023. doi:10.1186/s40537-023-00776-7.

[28] Cataldo Musto, Pierpaolo Basile, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Introducing linked open data in graph-based recommender systems. 53(2):405 – 435, 2017. doi:10.1016/j.ipm.2016.12.003.

[29] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. Knowledge graph convolutional networks for recommender systems. page 3307 – 3313, 2019. doi:10.1145/3308558.3313417.

[30] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. Explainable reasoning over knowledge graphs for recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5329–5336, 2019. doi:10.1609/aaai.v33i01.33015329.

[31] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. page 32 – 36, 2017. doi:10.1145/3109859.3109889.

[32] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), 2015. doi:10.1609/aaai.v29i1.9491.

[33] Peng Yang, Chengming Ai, Yu Yao, and Bing Li. Ekpn: enhanced knowledge-aware path network for recommendation. 52(8):9308 – 9319, 2022. doi:10.1007/s10489-021-02758-9.

[34] Malak Al-Hassan, Bilal Abu-Salih, Esra'a Alshdaifat, Ahmad Aloqaily, and Ali Rodan. An improved fusion-based semantic similarity measure for effective collaborative filtering recommendations. 17(1), 2024. doi:10.1007/s44196-024-00429-4.

[35] 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.

[36] 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.

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

[38] 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.

[39] Thiago Silveira, Min Zhang, Xiao Lin, Yiqun Liu, and Shaoping Ma. How good your recommender system is? a survey on evaluations in recommendation. 10(5):813 – 831, 2019. doi:10.1007/s13042-017-0762-9.

[40] Sonia Bergamaschi and Laura Po. Comparing lda and lsa topic models for content-based movie recommendation systems. 226:247 – 263, 2015. doi:10.1007/978-3-319-27030-2_16.

[41] Haebin Lim, Qinglong Li, Sigeon Yang, and Jaekyeong Kim. A bert-based multi-embedding fusion method using review text for recommendation. 42(5), 2025. doi:10.1111/exsy.70041.

[42] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[43] Iman Avazpour, Teerat Pitakrat, Lars Grunske, and John Grundy. *Dimensions and metrics for evaluating recommendation systems*. 2014. doi:10.1007/978-3-642-45135-5_10.

Appendix A: Plan of Work

Here is an overview of the work throughout the year 2024/2025 for the Bachelor's Thesis.

Week	Planned Work Description
Semester 1 – Research and Proposal	
Weeks 1–2	The topic will be selected and initial research on recommender systems will be conducted. Relevant literature on RS types, filtering techniques, and evaluation strategies will be reviewed.
Weeks 3–4	Detailed research and analysis of different types of recommender systems will be carried out.
Weeks 5–6	The recommender systems will be analyzed. The project scope will be defined, with a focus on the appropriate recommender method for digital library use cases.
Weeks 7–8	The evaluation methodology will be designed, including the selection of different metrics.
Weeks 9–10	Datasets will be collected: academic paragraphs and additionally other public datasets.
Weeks 11–12	A proposal will be written, outlining the implementation strategy for the book recommendation system and the evaluation plan.
Semester 2 – Implementation and Evaluation	
Weeks 1–2	A JSON parser will be implemented, and the data will be structured using Python. Experiments with various text representation models will be initiated.
Weeks 3–4	A recommendation pipeline based on cosine similarity and resource tracking will be developed. A Virtual Machine environment will be set up for conducting experiments.
Weeks 5–6	Offline experiments will be run on the datasets using all selected models, and outputs and similarity scores will be collected.
Weeks 7–8	Evaluation metrics will be computed. System performance will be logged (CPU, RAM, GPU usage). Visual outputs such as diagrams, radar plots, and deployment overviews will be finalized.
Weeks 9–10	The chapters on implementation, evaluation, and conclusion will be written, focusing on the interpretation of results.
Weeks 11–12	All chapters will be reviewed and finalized. Appendices and source documentation will be completed to finalize the thesis project.

Appendix B: Technical Documentation

The technical documentation was written in markdown format for the github repository and is attached as an appendix.
Here is the link to find the repository for the documentation, implementation and results:
github.com/Akos360/Evaluating_Recommender_Systems_Implementation

Appendix C: 2025 IIT.SRC Conference

Article created for IIT.SRC 2025: **Evaluating Recommender Systems for Digital Library Datasets**

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 Bať alík, 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].

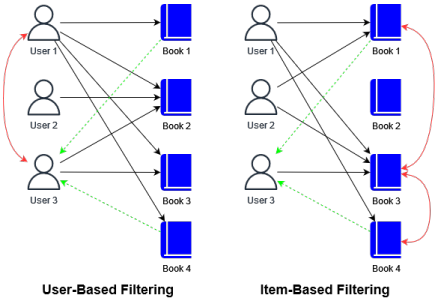

Fig. 1. Memory-based CF recommendation.

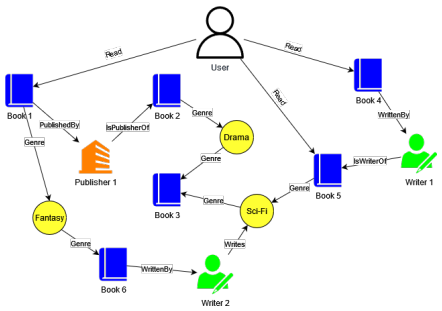

Fig. 2. KG-aware recommendation.

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

Appendix D: DEMOcon 2025

Article created for DEMOcon 2025: **Evaluating Recommender Systems for Digital Library Datasets**

Evaluating Recommender Systems for Digital Library Datasets

***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. 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.**

***Index Terms*—Recommender System, Content-Based, Evaluation, Performance Metrics, Digital Library**

I. 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 [10]. By analyzing large volumes of textual data, they aim to understand users' preferences and generate personalized recommendations [19]. 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 [16]. 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 [20]. 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 [18]. Recommender systems need to be adaptive in order to stay effective over long periods.
Major platforms like Netflix demonstrate the real-world impact of recommender systems by using a combination of personalized ranking algorithms, content similarity models, and short-term trend analysis to keep users engaged. Their system blends collaborative and content-based methods, optimized through continuous A/B testing and large-scale user data analysis [8].

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 [9], along with performance factors such as execution time or memory usage.

The rest of the paper is organized as follows. Section II provides a detailed overview of different types of recommendation systems, with a particular focus on content-based filtering approaches relevant to textual data. Section III presents the system design, including dataset construction, model selection, embedding strategies, and the deployment architecture of the recommendation system. Section IV outlines the experimental setup and reports the evaluation results across various metrics, comparing classical and modern embedding techniques on digital library content. Section V concludes the study with a summary of key findings and discusses potential directions for future research, including the integration of hybrid models and advanced transformer-based architectures.

II. Understanding Recommendation Systems

Recommending items can be done in a variety of ways. Several types of recommendation systems exist, and their methods of operation differ [17]. 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) [14]. This method is further divided into User-Based and Item-Based filtering which in short recommends items based on the users history or the items that the user interacted with.

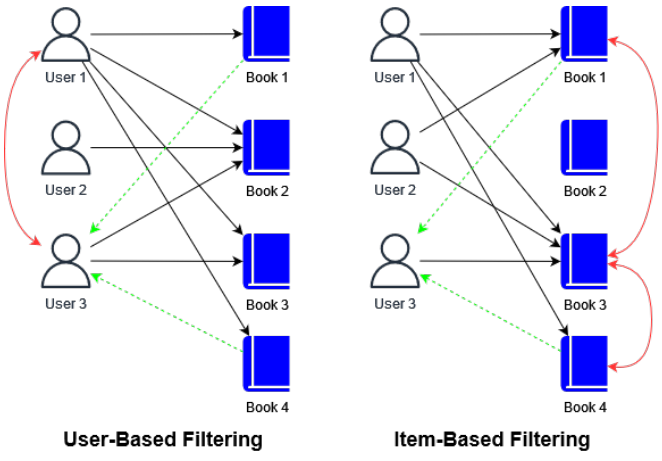

Fig. 1. Memory-based CF recommendation.

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 [11].

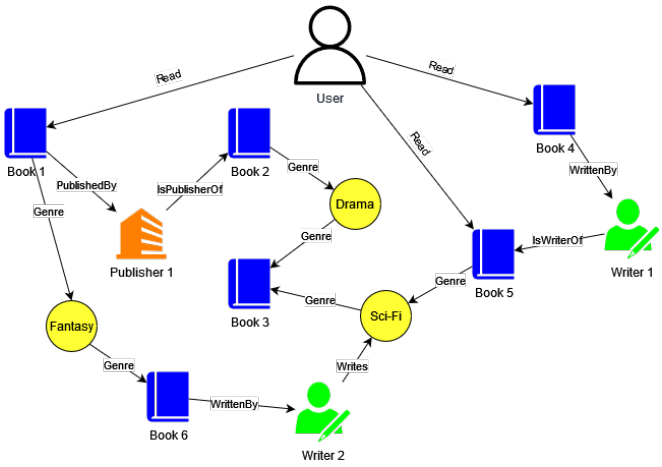

Fig. 2. KG-aware recommendation.

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 [10]. 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 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 [13].
The most popular techniques used are the Collaborative Filtering, Content-Based Filtering and the Hybrid method [5]. This paper will focus on comparing different algorithms 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.

III. 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. These JSON files were loaded and parsed into *Pandas DataFrames* for easier handling and segmentation into paragraphs. The processed data was then exported into CSV files and a dataset of 45,461 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 [12] 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** is a *pretrained static word embedding model*, where each word is represented by a fixed dense vector [15]. In contrast, **FastText** is a *trainable subword-aware model* that generates word vectors based on character n-grams, enabling it to compute embeddings even for out-of-vocabulary words. In this study, FastText was trained directly on the corpus to better adapt to domain-specific language.
- **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.

Deployment of the System

The following diagram illustrates the deployment architecture of the recommendation system. It highlights the main components involved, such as data ingestion from digital libraries, embedding generation, recommendation pipeline, and evaluation.

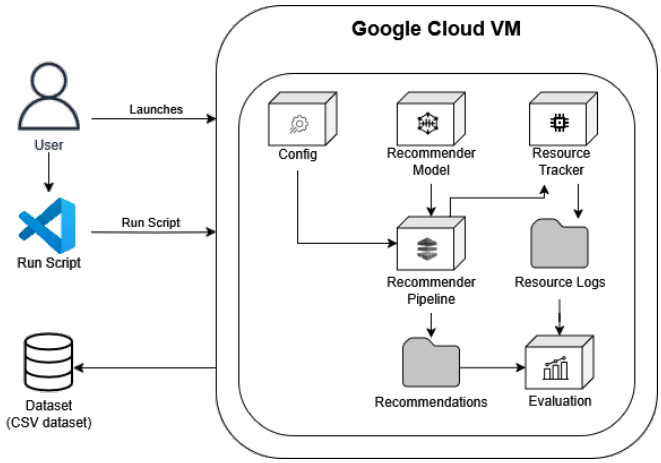

Fig. 3. Deployment Architecture of the Recommendation System

IV. 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 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.

While commonly used metrics like Precision, Recall, and F1-score are typical in evaluating recommendation systems, they rely on ground-truth user preference data such as ratings or clicks. Since this study is based on content-only datasets without explicit user feedback, such metrics were not applicable. Instead, we focused on intrinsic evaluation using similarity-based measures to reflect how well models capture textual relevance. The next table presents these results for paragraph-level inputs, which tend to be longer and more complex than book descriptions.

TABLE I
Performance on Book Descriptions

Metric	TF-IDF	BoW	FT	GloVe	LSA	BERT
Sim.	0.20	0.25	0.95	0.98	0.43	0.36
Cov. (%)	0.34	0.23	0.53	0.25	0.12	0.20
Conf.	0.95	0.94	0.99	1.00	0.93	0.95
Div.	0.0027	0.0050	0.0001	0.0000	0.0055	0.0033
Time (s)	3.97	15.26	49.96	14.88	13.24	74.46
Mem (MB)	293	378	2003	421	745	1227

While book descriptions offer a concise and curated summary of content, paragraphs extracted directly from books tend to reflect more diverse writing styles and granular topic shifts. Evaluating on paragraph data allows us to test how well each model adapts to natural, unstructured text as opposed to summary-based inputs. The next table presents the comparative results for this dataset.

TABLE II
Performance on Paragraph Dataset

Metric	TF-IDF	BoW	FT	GloVe	LSA	BERT
Sim.	0.28	0.19	0.94	0.99	0.30	0.35
Cov. (%)	0.06	1.30	15.32	21.52	1.20	9.29
Conf.	0.95	0.97	0.99	1.00	0.95	0.95
Div.	0.0027	0.0007	0.0001	0.0000	0.0027	0.0025
Time (s)	15.20	3.72	13.37	4.65	3.50	21.21
Mem (MB)	293	203	1193	344	425	1141

Tables I and II present the complete numerical evaluation results for each algorithm across the two datasets. These include quantitative comparisons based on similarity, coverage, confidence, diversity, processing time, and memory usage. By examining the exact values, readers can better understand how each method performs under specific criteria and identify trade-offs between computational efficiency and recommendation quality.

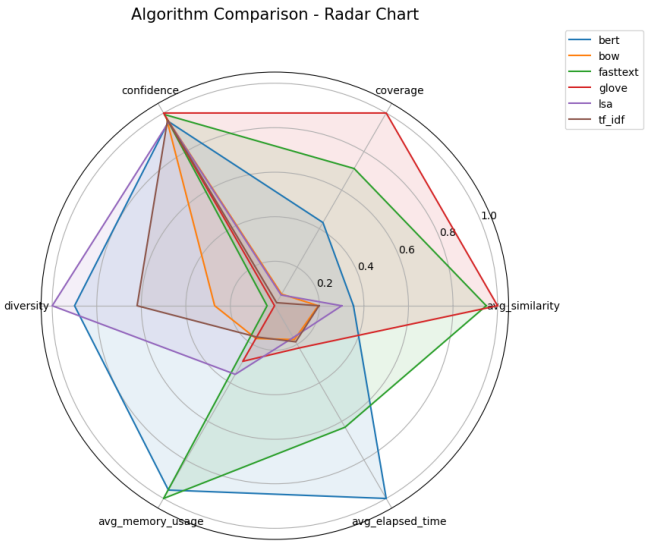

Fig. 4. Result of Metrics on Book Descriptions Dataset

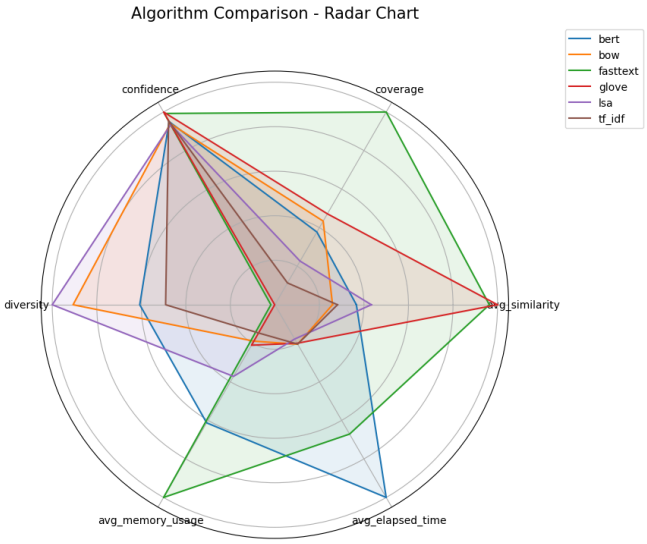

Fig. 5. Result of Metrics on Paragraphs Dataset

The radar charts (Figures 4 and 5) offer a visual summary of the performance metrics across all evaluated models. From the results, FastText and GloVe consistently achieve the highest similarity and confidence scores, confirming their strength in generating semantically close recommendations. However, they also show very low diversity and relatively high memory and time requirements. On the other hand, traditional models like TF-IDF and BoW are lightweight and efficient but perform worse in similarity and coverage. LSA presents a balanced trade-off, offering moderate similarity and strong diversity with reasonable resource use. Sentence-BERT shows competitive performance, especially in similarity and diversity, although at the cost of longer processing time. Overall, these comparisons help clarify the strengths and limitations of each model based on the specific needs of a recommendation task.

When comparing the two datasets, some notable differences emerge in model behavior. The book descriptions dataset generally results in higher average similarity scores across models, likely due to the more concise and topical nature of the descriptions compared to full paragraphs. Conversely, the paragraph dataset leads to higher coverage, especially for dense embedding models like GloVe and FastText, reflecting their ability to capture subtle semantic variations in longer and more diverse text segments. Additionally, the performance gap between traditional and neural models is more pronounced in the paragraph dataset, indicating that deep models may scale better with richer content structures, whereas simpler models are more suited for compact, well-defined summaries.

V. Conclusions and Future Work

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.

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 smaller simplified models, approximate nearest neighbor techniques, or compressed embeddings. Additionally, evaluating models across more diverse content types and multilingual datasets would help assess their generalizability and robustness.

REFERENCES

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

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

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] Carlos A. Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4), 2016. 10.1145/2843948.

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

[10] 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.

[11] 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.

[12] Deepak Kumar. Goodreads book data. https://www.kaggle.com/datasets/deepaktheanalyst/books-details-dataset, 2022. Accessed: April 12, 2025.

[13] 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.

[14] Mehrbakhsh Nilashi, Othman Ibrahim, and Karamollah Bagherifard. A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. 92:507 – 520, 2018. doi:10.1016/j.eswa.2017.09.058.

[15] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

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

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

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

[19] 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.

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

Appendix E: Digital Attachments

Here is the link to find the repository for the documentation, implementation and results:
github.com/Akos360/Evaluating_Recommender_Systems_Implementation