Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies

Ákos Lévárdy

Evaluating Recommender Systems for Digital Library Datasets

Bachelor thesis

Degree course: Informatics
Field of study: 9.2.1 Informatics
Place: FIIT STU, Bratislava
Supervisor: PaedDr. Pavol Baťalík
April 16, 2025

ANNOTATION

Slovak University of Technology Bratislava
Faculty of Informatics and Information Technologies
Degree Course: Informatics

Author: Ákos Lévárdy
Diploma Thesis: Evaluating Recommender Systems for Digital Library Datasets
Supervisor: PaedDr. Pavol Baťalík
April 16, 2025

This bachelor thesis focuses on analysing and evaluating recommender systems best suitable for digital library datasets. With the rise of digital content, these systems help users navigate large information spaces by generating personalized recommendations. In the introduction we showed the importance and role of Recommender Systems (RS) in filtering and anticipating user preferences across various domains. We then described the different recommendation techniques in detail, showing how they work and what their difficulties are. The project emphasizes Content-Based Filtering techniques, comparing algorithms like BERT, FastText, LSA, and TF-IDF. Offline experiments tested the system's ability to provide Top-N book recommendations from the input data and the algorithms were compared based on evaluation metrics such as Similarity, Coverage, Diversity, and Confidence. This work aims to evaluate different recommendation algorithms designed for digital library datasets.

ANOTÁCIA

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Študijný program: Informatika

Autor: Ákos Lévárdy
Diplomová práca: Evaluating Recommender Systems for Digital Library Datasets
Vedúci diplomového projektu: PaedDr. Pavol Baťalík
April 16, 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 sme ukázali význam a úlohu odporúčacích systémov pri filtrovaní a predvídaní preferencií používateľov v rôznych oblastiach. Následne sme detailne opísali rôzne techniky odporúčania, vrátane ich fungovania a výziev. Projekt kladie dôraz na techniky založené na obsahu (Content-Based Filtering), pričom porovnáva algoritmy ako BERT, FastText, LSA a TF-IDF. Offline experimenty testovali schopnosť systému poskytovať Top-N odporúčania kníh z vložených dát a algoritmy boli porovnávané na základe hodnotiacich metrík, ako sú podobnosť (Similarity), pokrytie (Coverage), rozmanitosť (Diversity) a dôvera (Confidence). Cieľom tejto práce je zhodnotiť rôzne odporúčacie algoritmy navrhnuté pre datasety digitálnych knižníc.

ACKNOWLEDGMENT

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

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 denote 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]

The main target of this project is to create a recommendation system that uses different algorithms for book recommendations and evaluate these algorithms based on specified metrics.

2 Understanding Recommendation Systems

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

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

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. Here are the different recommendation systems

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:

- 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 preferences 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. [4]

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]:

#	**Task**	**Description**
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.

Table 1: Recommendation System tasks

2.2 Recommendation Techniques

As mentioned before 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. 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

Recommender systems based on memory automate the common principle that similar users prefer similar items, and similar items are preferred by similar users [19].

Memory-based collaborative filtering, which can also be called Neighborhood-based 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.

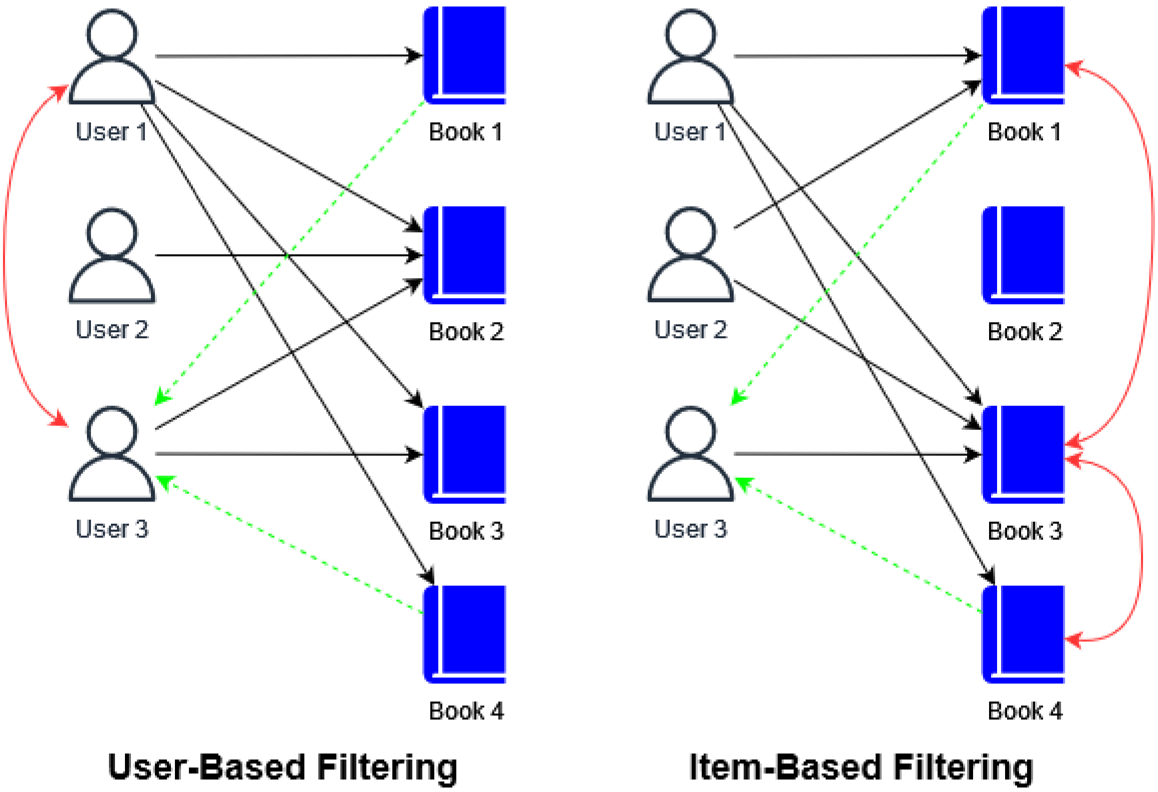

Figure 1: Illustration of Memory-based CF recommendation

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

Recommender 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, probabilistic Latent Semantic Analysis (pLSA) 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 predominantly contains additional knowledge about the recommendable items, e.g., in terms of their features, metadata, category assign-

ments, 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]

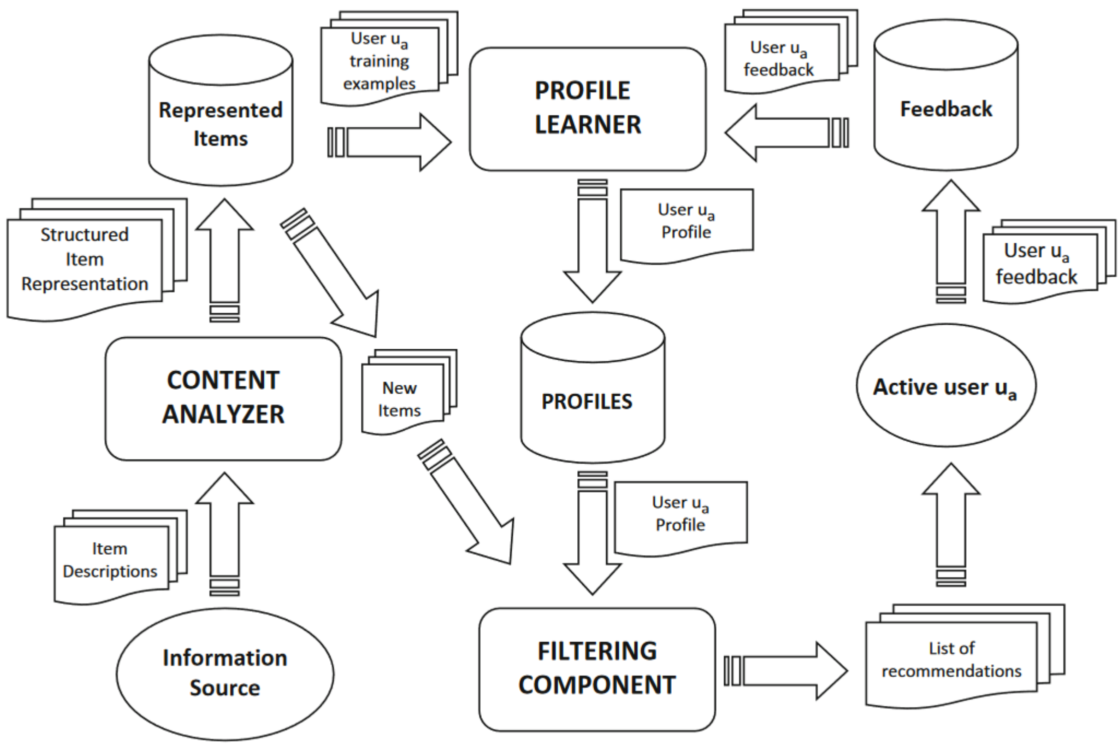

Figure 2: High level architecture of a content-based recommender [1]

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 user to build their own profile. Collaborative approaches will recommend items based on ratings 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 ratings 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, BabelNet) 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 representa-

tion 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 Word2vec and 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 an agreed semantics 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 databases 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].

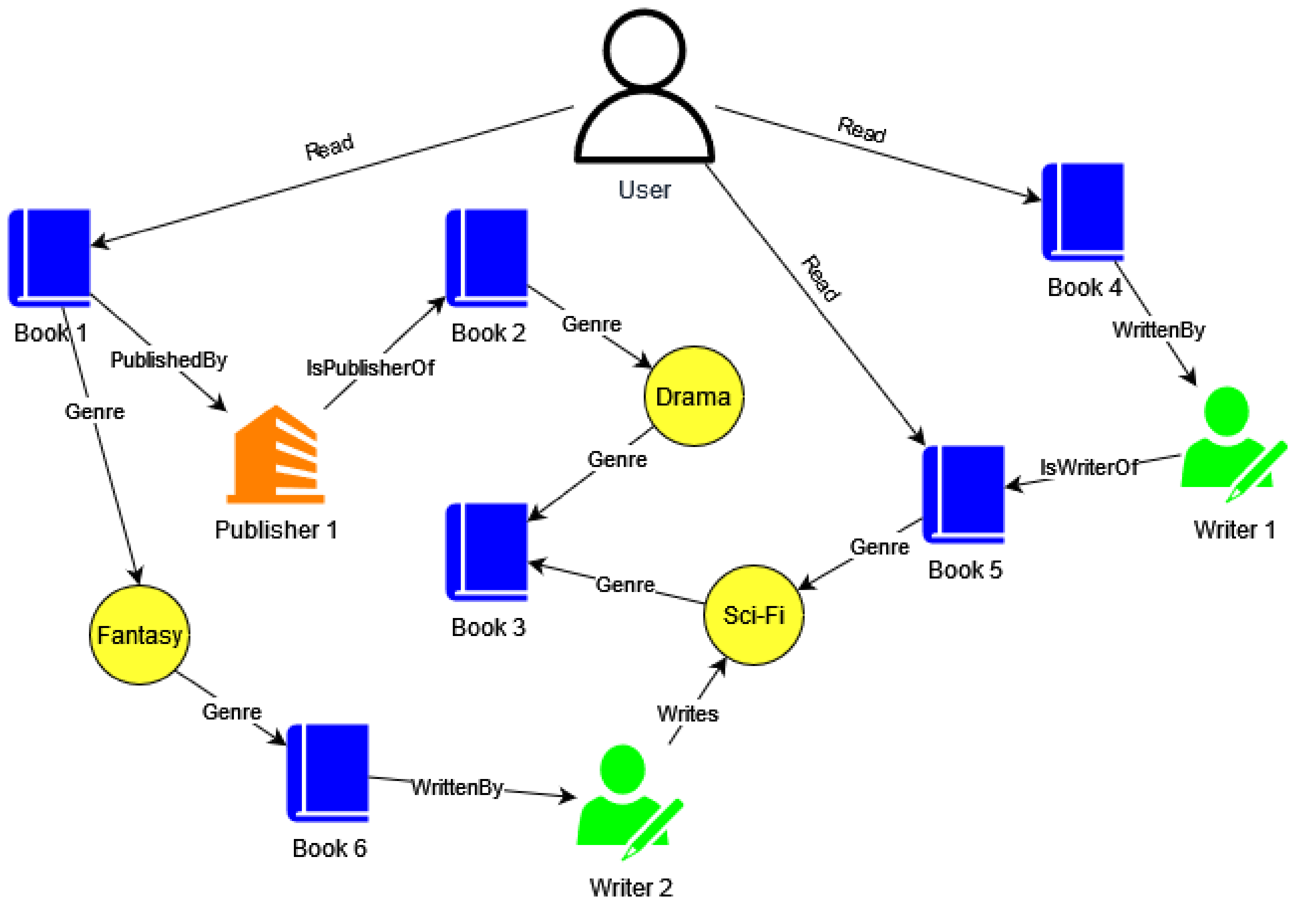

Figure 3: Illustration of KG-aware recommendation

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, which are the following [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** is not predicting an explicit rating, but rather is 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.

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 no advertising pushing new items, and the long tail of infrequently read articles may contain the so-called sleeping beauties, that are documents containing extremely relevant results, but that remains unknown to most researchers for a very long time.
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].
A study shows that more than half of the recommendation approaches applied Content-based filtering, when making recommendations for research papers and articles in libraries [22].

3 Implementation Proposal

Drawing from the theoretical foundation in the second section, this section proposes a practical framework for evaluating different Content-Based Filtering 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 choice 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 were converted from PDF format into structured JSON files, with the text segmented into individual sentences, paragraphs and pages. The experiments 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.

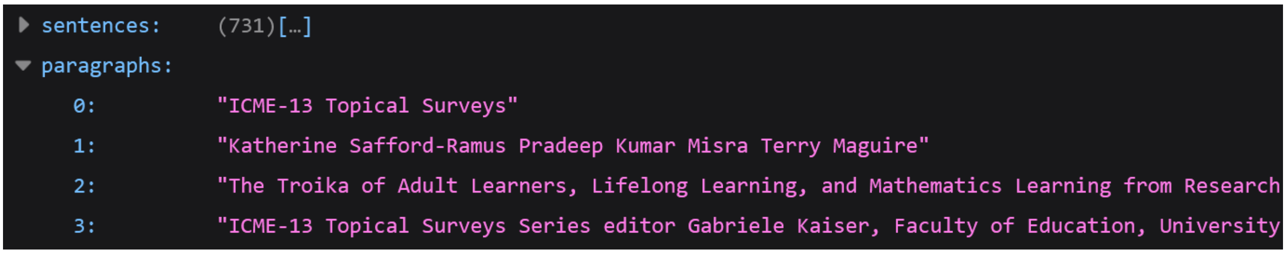

Figure 4: Example of extracted book paragraphs

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, scraped 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 ensures that the evaluation covers specialized and general-purpose recommendation use cases. Basic preprocessing such as tokenization, removal of special characters, and optional stopword filtering was applied prior to embedding.

3.2 Text Representation Methods

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

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

These models were chosen to represent a broad spectrum of natural language processing techniques, ranging from classical statistical approaches to modern deep learning-based embeddings.

There were 2 goals behind this selection. 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 were used for text representation, highlighting their methodology and key characteristics:

1. **TF-IDF (Term Frequency - Inverse Document Frequency)**
   TF-IDF was 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, while simple, provides a valuable lower bound for comparison. It does not consider word order and semantics, making it an 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 rela-

tionships 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. [39].

4. **GloVe (Global Vectors for Word Representation)**
   GloVe was 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 morphologically 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, implemented through the SentenceTransformers framework, 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.

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

3.3 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
    - BoW (Bag of Words)
    - LSA (Latent Semantic Analysis)

- GloVe
- FastText
- BERT (via 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: Similarity, Confidence, Coverage, and Diversity
   - Time and memory usage will be tracked to compare algorithm performance

The following diagram summarizes the overall flow of the proposed system:

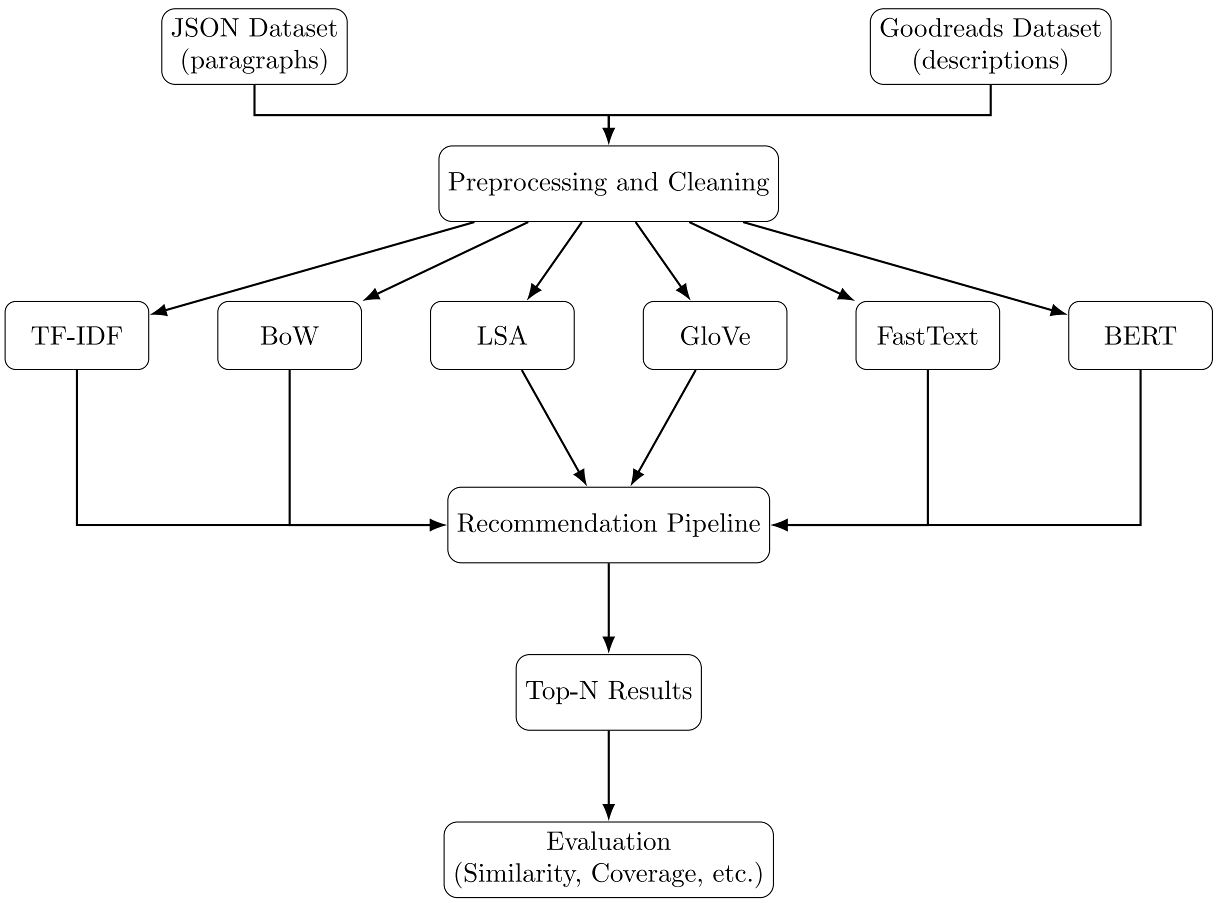

Figure 6: Conceptual flow of the proposed recommendation system with multiple text representation methods

4 Implementation

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

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. 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 [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 robustness and comparison.

4.2 Environment and Tools

Jupyter Notebook
Virtual Machine
Multiprocessing
GPU
Implementation was done in Python 3.12.6
libraries: ...

4.3 Used Text Represantation Models

4.3.1 Bert

```
def weighted_rating(R, v, m, C):
    return (v / (v + m)) * R + (m / (v + m)) * C
```

```
def weighted_rating(R, v, m, C):
    return (v / (v + m)) * R + (m / (v + m)) * C
```

4.4 Experiment Setup

The **Task** is to recommend Top N books (list of books) based on 1 input book (opened book).

Objectives:

- Implement and test multiple algorithms
- Compare performance of algorithms using evaluation metrics

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

- **Technical setup and execution**
  - Initial development will begin in Jupyter Notebook using Python
  - For multiprocessing and GPU support, the system will be deployed on a Google Cloud Virtual Machine
  - Multiprocessing will be used for CPU/GPU resource tracking

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 2: Overview of Experiment Types [3]

5 Results and Evaluation

This section presents how the system was evaluated using defined metrics and tracked in terms of performance.

Graphs

Tables of results

THRESHOLDS

BERT=0.5

BOW=0.5

FASTTEXT=0.95

GLOVE=099

LSA=0.5

TF-IDF=0.5

Description

5.1 Evaluation

The goal is to compare the previously mentioned algorithms 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, 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

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

Metrics for evaluation: It is crucial to define how Relevant an item is based on the input item and it will be calculated as how similar their features are. Higher similarity will show higher relevance between the compared items in the list of recommended items. The metrics below describe how to compare the algorithms and how they use relevancy between items:

- Similarity - between the vectors or embeddings
- Diversity - refers to the dissimilarity of the items recommended, where low similarity values mean high diversity. Ensures that the recommendations are not overly similar to each other and show a broader range of topics or genres. It is important to set a threshold for similarity between recommended items and the input item to make sure the recommendations are still sufficiently relevant. Firstly we need to define the attributes to assess diversity (topics, genres, keywords), then after the algorithms created the

feature vectors the Similarity Measures will show how dissimilar the books in the recommendation list are from one another. Then the overall diversity score is calculated from the recommendation list, also normalized.

- Confidence - shows the system's trust in its recommendations or predictions. It will be calculated based on Similarity between the Input item and the recommended items. After the algorithms created the feature vectors and the similarities are calculated from those, the confidence will be computed as the average similarity score across all recommended items, also normalized to a standard range [0, 1].

  A high confidence score indicates that the recommended items are very similar to the input item, suggesting the system is confident in its recommendations.

- Coverage - evaluates how effectively a recommendation algorithm utilizes the dataset, the proportion of items in the dataset that are recommended by the algorithm. High Item-space coverage indicates that the algorithm is exploring a wide range of items in the dataset and not favoring only a small subset. It is calculated as the percentage of: Number of recommended items / Number of total items in the dataset.

- Ratings can show prediction

DIVERSITY can be variance between the items and also can be cosidered as the contrary effect of similarity [40]

Also Genre Space Coverage can be mentioned

POPULARITY - for the ratings and reviews table or graph, define that the more popular items the user gets recommended the more satisfied the user will be, higher ratings -¿ higher value

NOVELTY -

An alternative approach for measuring novelty is to count the number of popular items that have been recommended [70]. This metric is based on the assumption that highly rated and popular items are likely to be known to users and therefore not novel [48]. 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 [26]. [41]

6 Conclusion

After implementing and testing the approaches of recommendation for books, I have found out that ...

7 Resumé

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

[40] 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. 10.1007/s13042-017-0762-9.

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