Department of Computer Science

Submitted in part fulfilment for the degree of BEng.

We Think You'd Like This: A Hybrid Recommender System for Goodreads

Madeleine Nielsen

28/04/2025

Supervisor: Dimitar Kazakov

Contents

Statement of Ethics

The ethical considerations and declarations for this project can be found in the project folder (Ethical Considerations Checklist and Declaration.pdf).

The Goodreads dataset used in this project was obtained from a public academic resource curated by the University of California, San Diego [1]. This dataset is anonymised, meaning it does not contain any personally identifiable information such as names, email addresses, or direct user identifiers. It was made publicly available for research purposes with clear documentation and usage terms. The data consists of user-item interactions, such as book ratings and reviews, which are aggregated and disconnected from individual identities. Since this project is theoretical and does not involve the direct analysis of real users or their personal data, there are no risks to privacy or data security.

Ethical concerns are often raised in the context of recommendation systems, particularly regarding privacy, data security, and the potential for reinforcing harmful biases. However, these concerns are not applicable in this project, as it does not involve the collection or analysis of sensitive data, and no real users are involved. The dataset used is explicitly anonymised, and no effort has been made to de-anonymise or link it to specific individuals.

In keeping with ethical research practices, the data was used strictly within the scope intended by its original release. All efforts were made to ensure compliance with the dataset's terms of use, and there is no concern regarding privacy breaches or personal data security in the context of this study.

Executive Summary

Recommender systems play a crucial role in helping users navigate vast digital collections by offering personalised suggestions. However, platforms like Goodreads face issues such as popularity bias, limited diversity, and lack of transparency. This project developed a hybrid system designed to address these challenges by balancing relevance, novelty, and explainability.

Traditional systems often prioritise popular content, narrowing exposure and failing to provide transparency in how suggestions are generated. They also perform poorly in sparse environments, offering subpar results for new or niche users. This system combines soft clustering, matrix factorisation, and graph-based learning to deliver accurate, diverse, and interpretable recommendations. The switching hybrid strategy adapts based on available user interaction data, as illustrated in Figure 2.2.

For users with minimal interaction history and new users, the system generates recommendations by using HDBSCAN to group previously read books based on feature similarity. This helps uncover niche content, encourages exploration, and reduces the reliance on popularity-based suggestions. Singular Value Decomposition is used to impute ratings where a user has read a book but no rating is available in the dataset. This helps complete sparse user profiles and improves overall recommendation quality across the system.

Users with a moderate level of interaction are supported by Non-Negative Matrix Factorisation. This model captures patterns in user preferences and relationships between items. Its non-negativity constraints support transparency and produce interpretable recommendations. For highly active users, the system switches to a graph-based model using GATv2Conv combined with Non-Negative Matrix Factorisation. GATv2Conv combines user-item interactions with sentiment and genre data to give more personal recommendations. The NMF and GATv2Conv outputs are ranked and weighted to ensure a balance of precision and diversity. This hybrid model makes sure that recommendations reflect individual tastes while also drawing from broader patterns in the data.

The system was evaluated using a comprehensive set of metrics to ensure a complete assessment. The root mean square error and the mean absolute error were used to gauge the predictive accuracy by seeing how well the system predicted the ratings for user interactions with the books. To assess the quality of the recommendations in terms of classification, normalised discounted cumulative gain and mean reciprocal classification focused on

how highly relevant items were ranked in comparison to less relevant ones. Given the project's emphasis on discovery, novelty and diversity ratio were key priorities in the evaluation process of the hybrid pipeline.

These metrics made sure that the system not only made accurate predictions but also encouraged users to explore new, diverse content beyond the most popular items, aligning with the project's goals of reducing popularity bias and enhancing user engagement through varied and meaningful suggestions.

The hybrid recommender pipeline successfully recommended all genres in the dataset so that recommendations were not concentrated in a narrow subset of popular categories. The average similarity score for HDBSCAN-based recommendations was close to 1 showing strong internal coherence within clusters. This reinforced the system's ability to balance diversity with relevance, supporting its goal of offering engaging and novel content. HDBSCAN was assessed using Davies-Bouldin, Calinski-Harabasz, and Silhouette scores. These clustering scores showed good performance, showing that the groups of books formed were compact and well separated. In addition to its quantitative strength, HDBSCAN also proved highly interpretable. Since it groups books based on shared features, users can more easily understand recommendations, this transparency was a key part of the systeInterpretability

Interpretability varied across the hybrid pipeline. Non-Negative Matrix Factorisation provided moderate explainability due to its use of latent factors so users can infer general themes or preference patterns. In contrast, the GATv2Conv model performed well in generating personalised and diverse suggestions, but offered limited interpretability due to the complexity of attention mechanisms and the integration of multiple data types. This trade-off between depth of personalisation and transparency highlights one of the main challenges in designing advanced recommendation systems.

The hybrid system successfully balances accuracy, novelty, and diversity by combining collaborative filtering, graph learning, and clustering to counteract popularity bias, support cold-start users, and broaden the scope of recommendations. Clustering was able to surface rare books and making system suggestions more understandable.

One limitation is the lack of implicit user feedbackfeedback,luding this could make recommendations more adaptive and responsive. This project demonstrates that a ranked weighted hybrid recommender within a switching framework can overcome many of the limitations faced by traditional systems. Although the results are promising, more work is needed to reduce system complexity, improve interpretability in deep models, and deliver more interactive, context-aware recommendations.

Introduction

Recommender systems play a critical role in navigating large volumes of digital content, helping users discover items that match their interests and preferences. Since the 1990s, these systems have been developed to improve content discovery and user engagement by analysing behavioural data and preferences. As digital collections expand, platforms like Goodreads now host millions of books, making manual selection impractical. Users engage with only a fraction of the available items, highlighting the need for algorithms that assist in surfacing relevant content.

Motivation: Manual selection and bestseller lists tend to favour popular books, overlooking lesser-known titles and limiting exploration. A modern recommender system should balance all of this by offering suggestions aligned with users' interests while promoting discovery. Despite its popularity, Goodreads struggles to provide personalised and transparent recommendations due to its outdated design and popularity-driven algorithms. Traditional collaborative filtering struggles with cold-start users who lack interaction history, and content-filtering can enforce bias and limit exploration. To address these issues, this project explores a hybrid recommendation system designed to be more accurate, novel, and transparent, even with sparse data.

1.1 Aims and Evaluation Criteria

Aims

- Encourage exploration by recommending both popular and less known books.
- Handle cold-start users by not relying on user interaction history.
- Provide explainable, transparent recommendations.
- Personalise recommendations to align with user's past preferences.
- Build a system that balances accuracy, novelty, and user trust.

Evaluation Criteria

- Achieves a balanced proportion of recommended books that are less known or less popular.
- Successfully recommends relevant books for users with limited interaction history.
- Provides clear and interpretable explanations for recommendations.
- Aligns recommendations with users' past preferences and reading behaviour.
- Demonstrates strong performance across ranking, accuracy, and discovery metrics.

Literature Review

This section covers the methods and models implemented in modern recommender systems, focusing on collaborative filtering, content-based filtering, deep learning, and hybrid techniques along with their relevance to improving recommendation accuracy. Modern techniques such as deep learning and graph-based models have been researched to see how to improve the personalisation and transparency of recommendations. The evaluation of these models covers technical performance, ethical, and fairness aspects, which are all needed for building trustworthy systems.

2.1 Content-Based

Content-based filtering recommends items by analysing their feature [2] and identifies items that are similar to those a user has previously interacted with, based on these features. Similarity is often measured using techniques like Cosine similarity or distance metrics applied to feature vectors. While this approach can deliver highly relevant suggestions, it often leads to a 'filter bubble' [3], where users are exposed only to items similar to their previous choices. This over-specialisation [4] can limit diversity and reduce opportunities for discovering new genres or authors. Content-based methods are effective for addressing the cold-start problem for new users as they do not rely on the preferences others. They also provide transparent recommendations, as the features influencing the suggestions are easily explained and traced.

Clustering can enhance content-based filtering by grouping items with similar features [5], reducing the search space and improving recommendation efficiency [6] as recommendations can focus on items within the same cluster, improving similarity calculations and handling high-dimensional data and sparsity. Shukla and Chowdary [7] address the cold-start problem in peer recommendation by applying k-means clustering on authors' publication features and doc2vec embeddings to rank clusters. Their model combines clustering with attribute-based filtering to identify relevant collaborators, improving recommendation relevance and depth.

HDBSCAN extends DBSCAN [8] by using a hierarchical clustering approach that detects clusters based on density without requiring a predefined number of clusters [9]. HDBSCAN identifies dense regions and merges or splits clusters based on density thresholds, forming a hierarchy that can be flattened. For instance, books on dystopian themes, space exploration,

and science fiction may form distinct clusters, while books with similar text but no explicit genre labels can also be grouped. HDBSCAN's ability to detect clusters of varying densities makes it ideal for tasks involving diverse genres and overlapping themes. Performance heavily depends on selecting the distance metric, such as Cosine or Euclidean distance.

2.2 Collaborative Filtering

User-based collaborative filtering suggests items based on the preferences of similar users. It calculates user-user similarity by comparing ratings or interactions and creates a user-user similarity matrix [10]. The ratings of the most similar users are averaged to predict a user's rating for items they haven't rated. This method uses the collective preferences of the user base to offer personalised recommendations, making it easier to discover new items. It focuses on user behaviour rather than item features, potentially making the recommendations more personal and novel[11]. However, it faces challenges like the cold-start problem for users or items with limited data, and is computationally expensive due to the need to compare every user to every other user[12].

Model-based collaborative filtering uses machine learning algorithms to predict user preferences by learning patterns from historical interaction data. They scales better than memory-based methods on large datasets and often provide more accurate predictions. However, these models can suffer from popularity bias and may struggle with the cold-start problem for new users or items without prior interactions.

Model-based filtering learns user and item representations to predict ratings for unseen pairs [13]. The winning team of the Netflix Prize used a combination of SVD models, including SVD++, alongside Restricted Boltzmann Machines. By blending these techniques, they achieved a 10% improvement in accuracy over Netflix's existing algorithm [14]. SVD decomposes the user-item matrix ($\mathbf{A}$) into three matrices: one representing relationships between users and latent factors, another containing singular values indicating the importance of each latent factor, and one representing relationships between items and latent factors [15].

The user-item matrix is often sparse, so low-rank approximation simplifies it by retaining significant components, which is more efficient. A randomised approach can be used for matrix splitting to speed up the process, making it suitable for recommender systems where approximations are acceptable. However, traditional SVD is computationally intensive, requiring significant memory and processing power. The handling of missing values, like

imputing zeros or averages, can significantly affect recommendation quality.

NMF [16] is a dimensionality reduction technique widely used in recommender systems. It decomposes the user-item matrix $\mathbf{A}$ into two non-negative matrices: $\mathbf{W}$, which represents the user-latent factor relationships, and $\mathbf{H}$, which represents item-latent factor relationships. NMF enforces non-negativity, providing a parts-based representation that is easier to interpret. As the user-item matrix $\mathbf{A}$ is typically sparse, NMF approximates this matrix as $\mathbf{A} \approx \mathbf{WH}$, and so is efficient for large datasets, can be distributed, and keeps important components by reducing data complexity and filtering noise. NMF is valued for its interpretability and ability to manage sparse datasets efficiently. The paper "Large-scale Matrix Factorization with Distributed Stochastic Gradient Descent" explores scalable implementations of NMF, showing how effective it can be when handling large-scale recommendation tasks [17].

2.3 Deep Learning Models

Recent advancements in deep learning in fields like computer vision and speech recognition, have significantly improved recommendation systems. Research has shown that deep learning can substantially enhance a recommender system's performance compared to traditional models. As LeCun, Bengio, and Hinton [18] note, deep learning has been transformative across various domains, and its application in recommendation systems is no exception.

GATv2Conv [19] is an advanced version of GATConv [20], designed to address the limitations of static attention coefficients in GATConv. It uses attention mechanisms to assign varying importance to neighbouring nodes, allowing the model to focus on the most relevant parts of the graph structure [21]. By applying the attention mechanism directly to the concatenated node embeddings before the LeakyReLU activation, GATv2Conv improves the representation of nodes and better captures their relationships. The attention coefficients control the influence of neighbouring embeddings in updating node embeddings.

GATv2Conv is capable of handling multidimensional edge features and excels in node classification, link prediction, and recommendation systems. Its flexibility and scalability make GATv2Conv a powerful tool for various graph-based systems, letting it outperform traditional GCNs in many scenarios [22]. Graph-based models like GATv2Conv can be difficult to make explainable, as the learned attention weights and embeddings often lack clear, human-interpretable meaning, posing a challenge for transparency

and user trust in recommendation systems. GATv2Conv and other deep learning models can be made explainable by using techniques like Grad-CAM [23], LIME [24], or SHAP [25]. These methods highlight important features influencing the model's decisions, offering insights into its reasoning. However, these are hard to interpret into real, understandable features, requiring careful analysis to connect them back to the original data.

2.4 Hybrid Systems

Hybrid recommendation systems aim to improve the quality and robustness of recommendations by combining multiple recommendation techniques, each with its unique strengths. These systems address the limitations of individual models, such as the cold-start problem in collaborative filtering or the lack of diversity in content-based filtering.

Weighted hybrids combine the outputs of multiple models by assigning weights to their predictions to help balance the strengths of different techniques. The weighted hybrid system in the paper 'Weighted hybrid technique for recommender system' by S. Suriati et al [26] struggles to improve on error but it does help to give prediction scores for unrated items that could have been impossible to be recommended by only using collaborative filtering.

In ranked-weighted hybrids, multiple models generate ranked lists of recommendations and the top recommendation from each model is ranked 1st, then 2nd, etc. A weight is then applied like above so the best recommendations are prioritised. This allows for fallback models to generate additional recommendations when the top model's list is sparse. Burke [27] shows how this method improves accuracy and coverage by combining results from multiple models, prioritising better-performing ones while maintaining diversity. This technique is effective in e-learning and e-commerce [28], where user satisfaction and robustness against sparse data is needed.

Switching hybrids dynamically select which model to use based on certain conditions, such as the user's activity level. If a user has sufficient ratings, the system may switch to collaborative filtering, while for new users, it may fall back on content-based filtering. DailyLearner, an online news recommender [29], uses a short-term content-based filtering recommender which considers the recently rated news stories using Nearest Neighbor text classification and Vector Space Model with TF-IDF weights. If a new story has no near neighbour, the system switches to the long-term model, which is based on data collected over a longer period.

2.5 Similarity Calculations

Cosine similarity 1.13 measures the angle between two vectors, focusing purely on direction and ignoring magnitude. This makes it effective for identifying items with similar patterns or preferences, regardless of scale [30]. It is especially useful in text-based and embedding-driven recommendation systems, where relative orientation carries more meaning than raw feature values. However, by ignoring vector length, it may overlook intensity or strength of preference, making it less suitable when magnitude holds semantic value [31].

FAISS (Facebook AI Similarity Search) is a library optimised for efficient similarity search in high-dimensional spaces, making it suitable for large-scale recommender systems [32]. It supports both exact and approximate nearest neighbour (ANN) searches. The approximate methods are preferred for real-time recommendations at scale despite some precision loss [33].

Dot product 1.14 similarity calculates the alignment between two vectors, focusing on their direction rather than magnitude. A higher dot product indicates stronger similarity, especially in embedding-based models [34]. It is computationally efficient and scales well with sparse and high-dimensional data, making it ideal for real-time collaborative filtering. However, it assumes that larger vector norms imply higher importance, which may not always reflect true semantic similarity [30]. Euclidean distance 1.15, in contrast, measures the absolute difference in position between vectors in space, considering both direction and magnitude. A smaller Euclidean distance means higher similarity. This is useful when absolute feature values matter, such as when comparing user profiles or item features with clear thresholds (number of pages, average rating). However, its performance may degrade in high-dimensional spaces due to the increased sparsity of data and the challenge of distinguishing meaningful distances between points [31].

2.6 Evaluation Metrics & Techniques

Ratings prediction evaluates how accurately a recommender system predicts the ratings users would assign to items. The dataset is typically split into training and test sets. Common metrics include Mean Absolute Error (MAE) 1.1 and Root Mean Squared Error (RMSE) 1.2. MAE measures the average absolute difference between predicted and actual ratings, with lower values indicating better performance [35]. RMSE calculates the

square root of the average squared differences, placing greater emphasis on larger errors and making it more sensitive to significant deviations [36].

The Netflix Prize used RMSE as the primary evaluation metric, but this assumes that missing data is random, which may not hold in practice. Srebro and Salakhutdinov [37] demonstrated that users tend to rate familiar items higher, indicating that data is not missing at random. As a result, RMSE may not accurately reflect true prediction performance. Ranking evaluation metrics assess how effectively a recommender system orders items by relevance, comparing predicted rankings with actual user preferences. In practice, ranking metrics are often more informative than rating-based evaluations [38], as they focus on the relative position of items in the recommendation list, which is needed for improving user satisfaction.

A crucial element in ranking metrics is setting a relevance threshold, which converts predicted scores into binary relevance (relevant if above the threshold) and affects the precision-recall trade-off [39]. The choice of K, the number of items considered in the evaluation, is critical, as it reflects real-world constraints like the number of recommendations displayed. Precision@K 1.3 evaluates the proportion of relevant items among the top K recommendations, while Recall@K 1.4 measures the fraction of all relevant items retrieved in the top K. nDCG@K (Equation 1.5) compares the system's ranking to the ideal ranking (IDCG), with higher values indicating that relevant items are ranked more effectively. Mean Reciprocal Rank@k (mRR) 1.6 averages the reciprocal ranks of the first relevant item, with higher mRR showing better performance in placing relevant items early. Mean Average Precision@K (mAP@k) 1.7 is an advanced version of Precision@K, offering a more holistic measure of precision by averaging the precision across multiple values of K, considering the precision at each ranked position up to K. [40].

2.6.1 Beyond Accuracy

Recent advancements in recommender systems have highlighted the need for more appropriate evaluation metrics, as traditional measures may not fully capture the capabilities of emerging models.These metrics assess not only accuracy but also the ability to deliver a personalised and varied user experience [41]. Proper recommender system evaluation needs to include prediction accuracy and also their ability to increase user engagement.

Novelty 1.12 is an essential property for enhancing user satisfaction and reducing the risks of filter bubbles. It reflects the degree to which recommendations present less popular items so the system can expose users to content they might not have encountered otherwise [42]. Vargas and

Castells [43] proposed a novelty-aware ranking framework that balances popularity and novelty, demonstrating its effectiveness in improving user retention and satisfaction over time. Hit Rate at K 1.8 gives an intuitive measure of system effectiveness by calculating the share of users for whom at least one relevant item is present in the top K recommendations [44]. User Coverage 1.9 measures the proportion of users for whom the system provides at least one relevant recommendation within the top K items, indicating the model's ability to cater to diverse user needs [45]. Item Coverage 1.10 assesses the proportion of items recommended to users across all recommendations, reflecting how well the system distributes recommendations across the catalogue [46]. Diversity Ratio 1.11 measures the variety of items in the recommendations, ensuring that users are exposed to a broad range of items [47].

2.6.2 Evaluating Without Ground Truth

In the absence of explicit user feedback, content-based filtering systems may rely on clustering to generate recommendations based on item characteristics, such as genre, metadata, or text embeddings. Without a clear ground truth to define "similarity," evaluating these systems becomes more interpretive and often depends on how well the clustering aligns with human-understandable patterns. Clustering quality can be evaluated using several metrics:

- **Silhouette Score** measures cluster cohesion and separation, with higher values indicating better-defined clusters [48].
- **Calinski-Harabasz Index** is the ratio of the sum of between-cluster dispersion and within-cluster dispersion. Higher values suggest that data points are more spread out between clusters than they are within clusters [49].
- **Davies-Bouldin Index** assesses the average similarity between clusters, where lower values indicate better separation [50].

Measuring the cosine similarity between item embeddings can reveal how semantically coherent items are within clusters. Topic modelling techniques like Latent Dirichlet Allocation [51] can help surface common themes, and evaluating topic coherence scores provides insight into how meaningful and interpretable the clusters are from a thematic perspective.

2.7 Transparency & Explainability

Recommender systems (RSs) have traditionally prioritised accuracy [52], but challenges such as attacks, noise, and biases highlight the need for the development of Trustworthy Recommender Systems (TRSs) [53]. These systems strive to balance prediction accuracy with ethical considerations, such as fairness, transparency, and robustness. Building TRSs requires ensuring high data quality and reducing biases, safeguarding user privacy, and improving model interpretability. Key methods for enhancing TRSs include :

- **Fairness**: Fairness in RSs is critical due to their widespread use in decision-making. Hua et al. [54] proposed UP5, a fairness-aware model using counterfactually fair prompting (CFP) to mitigate biases, ensuring recommendations don't disproportionately favour certain user groups based on attributes like gender or age. Zhang et al. [55] further developed fairness metrics and datasets, evaluating ChatGPT across sensitive attributes. Hou et al. [56] formalised recommendations as conditional ranking tasks to improve fairness in item exposure.

- **Ethical Evaluation**: Ethical assessments ensure recommendations align with societal welfare by incorporating fairness, transparency, and accountability. Methods such as content-based filtering and adherence to ethical guidelines help mitigate harmful biases and promote inclusivity, ensuring that recommendations support positive societal impacts [57].

- **Interpretability**: Interpretability is essential in recommender systems, as it builds user trust, aids decision-making, and increases system transparency by revealing the rationale behind recommendations. Zhang and Chen [58] introduced natural language explanations to improve transparency. Cui et al. [59] enhanced explanation quality by incorporating item features into prompts. Liu et al. [60], [61] advanced interpretability using continuous prompt vectors and demonstrated the effectiveness of ChatGPT in generating coherent explanations.

A comprehensive evaluation of TRSs should combine technical performance metrics and ethical considerations to ensure recommendations align with user preferences and societal well-being. Advances in interpretability and fairness are key to building trustworthy systems.

Problem Analysis

3.1 Data

Goodreads is the world's largest platform for readers and book recommendations, founded in December 2006 by Otis and Elizabeth Chandler. It is a social cataloguing website that allows users to search its extensive database of books, ratings, and reviews. Users can register books to create reading lists, groups of book suggestions, surveys, polls, blogs, and discussions. Goodreads' recommendation system often faces criticism for lacking personalisation, suggesting popular books instead of ones more aligned with users' interests [62]. Users have also expressed concerns about the lack of transparency in how the algorithm works, making it hard to understand the criteria behind suggestions [63]. Additionally, some users have reported receiving duplicate recommendations or suggestions for books they've already read or rated [64].

The Goodreads book graph datasets [1] were collected in late 2017 from Goodreads by scraping users' public shelves (available without login) by a team at the University of California, San Diego [65] [66]. Goodreads' terms of service restricts scraping the platform and automated access may result in bans or legal issues, so it is not possible to easily collect newer data directly from the site. So, this older dataset was selected for its scale, detail, and relevance. It remains one of the few publicly available, large-scale datasets that support collaborative and content filtering approaches within a real-world book recommendation setting.

Key Features:

- **Ratings:** Users rate books (1-5 stars).
- **Reviews:** Users write text reviews (up to 20,000 characters).
- **Custom Shelves:** Users categorise books (e.g., "Currently Reading", "Want to Read").
- **Social Interactions:** Users vote and comment on reviews.
- **Title:** Useful for identification and semantic analysis.
- **Summary:** Descriptions from back covers, publishers, or Wikipedia.
- **Authors:** List of all authors.
- **Genres:** Extracted from popular user shelves via keyword matching.
- **Cover URL:** Link to the book's cover image.
- **Similar Books:** List of similar books (IDs).

3.2 Methodology

To overcome the shortcomings of traditional recommender systems, the project will use a switching hybrid framework that adapts to different levels of user engagement. Instead of relying on complex methods which are unsuitable for the static dataset and outside the project's scope, the system will use simpler models that remain interpretable, efficient, and scalable. By selecting appropriate recommendation strategies based on user profiles, it will generate relevant suggestions for both new and highly active users.

To address missing rating data (where a user has read a book but no rating is present in the dataset), the system will use SVD (Surprise library [13]) to predict ratings for books users have read but not rated. Implicit Alternating Least Squares [67] is better suited for implicit feedback but SVD is more appropriate here as it can generate explicit ratings that can be used in the recommendation process.

To address cold-start users, the system will rely on content-based filtering using HDBSCAN (HDBSCAN library [9]). Figure 2.4 illustrates the distribution of books read per user, showing a peak at the beginning followed by a rough bell curve. This distribution indicates that while many users are engaged, a significant portion have limited interactions, justifying the need for content-based filtering. HDBSCAN does not require a predefined number of clusters, unlike K-Means [68]. It can also identify outliers, which is useful for highlighting niche books that might be overlooked, reducing popularity bias and increasing diversity in recommendations.

Memory-based collaborative filtering would not be suitable to predict ratings in this context as the dataset is very sparse, with many missing ratings. NMF (Surprise library [13]) can generalise more effectively across sparse data by learning latent patterns from the available interactions. The non-negativity constraints in NMF ensure that the resulting factors correspond to real-world concepts, which enhances interpretability. NMF also allows inspection of the latent features contributing to each prediction, making recommendations more transparent and explainable.

The system will use also use GATv2Conv to predict ratings(PyTorch [19]) as although GraphSAGE [69] can model graph-structured data by aggregating information from neighbouring nodes, it relies on fixed functions that treat all neighbours equally. GATv2Conv assigns dynamic weights to edges, prioritising more relevant relationships making it suited to heterogeneous graphs where nodes contribute unequally to predictions. Although Gatv2Conv may be less interpretable than simpler models, it better captures intricate user-item interactions and can incorporate other data such as embeddings.

Design

Figure 2.1 shows how the imputed ratings from SVD are incorporated into the main system and how different data sources feed into the individual models. SVD and NMF will reply on ratings, and user and book IDs. HDBSCAN will use embeddings generated from book features and GATv2Conv will use rating data, user and book IDs, and review data. Figure 2.2 shows the hybrid pipeline, where the appropriate recommendation strategy is based on the user's history of ratings and reviews to balance inclusivity and accuracy by applying simpler models for less engaged users and more complex models for highly engaged users.

The HDBSCAN recommender for new users tackles the cold-start problem by sampling from clusters to try and avoid popularity bias. When a new user arrives their preferences can't be inferred so a subset of clusters is randomly selected to ensure variety and expose the user to diverse genres. For users with fewer than 5 ratings, HDBSCAN will find which clusters the books they have read belong to and recommend based on the nearest items.

For users with more than 5 ratings, the system will use NMF (Surprise library [13]) to predict ratings for books users haven't read by uncovering latent features of users and books. For users with at least 5 ratings and 5 reviews, the system will use GATv2Conv to predict ratings (PyTorch [19]), as it is particularly suited for users with sufficient engagement data because it captures complex, non-linear relationships between users and items. GATv2Conv uses an attention mechanism to weigh the importance of different user-item relationships, making it more effective in personalising recommendations.

Figures 2.6 and 2.5 provide further insight into the distribution of user engagement. Most users have rated fewer than 50 books, with the largest group falling in the 21–50 and 51–100 ratings range. For reviews, the distribution is sparser, with most users contributing fewer than 10 reviews. This indicates that while some users are highly active, a large proportion have limited interactions. Based on these distributions, the system sets a threshold of 5 ratings and 5 reviews to ensure reliable recommendations for users with sufficient data while still accommodating users with fewer interactions.

The evaluation will use a range of metrics (see Tables 6.1, 6.2, 6.3) to provide a balanced view of accuracy, relevance, and diversity. This helps the system recommend items that match user preferences, support discovery, and promote broader engagement.

Implementation

The full code base can be found in the project folder and here, this repository contains details on how to get the data needed for the project [1].:
https://github.com/mud66/Recommender-System-GoodReads The implementation was carried out with an AMD Ryzen 7 5800U processor with Radeon Graphics, with 8 cores, 16 logical processors, and 16GB of RAM. The code was developed in Jupyter Notebooks for the interactive development environment.

5.1 Preprocessing

The 'similar books' column was excluded from this project as the dataset does not specify the criteria for determining book similarity, so its reliability and relevance couldn't be assessed. For large datasets like the 'User-Book Interactions' dataset, using the entire dataset is often impractical due to computational limits, so the first 1 million rows of the books, reviews, and interactions datasets were used in the system.

The data includes a "custom shelves" column listing the shelves a book has been added to. Many shelves are generic (e.g., "to-read," "read"), offering limited context. To reduce bias, these are excluded and column is expanded to reflect the frequency of the shelves.

The "language code" column in the Book Graph dataset identifies book languages, but many values are missing. To simplify, only books with English language codes or missing or invalid codes are kept (' ', 'eng', 'en-US', 'en-GB', '–', 'en-CA', 'en-IN'). The Lingua library's LanguageDetectorBuilder [70] is used to verify that only English reviews are processed. Book descriptions are then merged with authors, common shelves, and genres, reducing feature engineering complexity. Research shows that readers prioritise descriptions, authors, and ratings [71], which supports focusing on these features. These are embedded using the SentenceTransformer model [72], all-MiniLM-L6-v2 [73], trained on English text. While this introduces bias by excluding non-English books, it ensures consistent data quality and avoids distortion.

[1] The original datasets can be found here: https://cseweb.ucsd.edu/~jmcauley/datasets/goodreads.html and the sampled and processed data used for training and the models can be found here https://drive.google.com/drive/folders/1IME8HPKDIt1SNNqZM4rrcO8ERoihCjAC?usp=drive_link

Goodreads allows users to rate books on a 1-5 star scale but excludes half-star ratings. The 'User-Book Interactions' and 'Complete Book Reviews' datasets include '0' star ratings, indicating a user has read a book but left it unrated, as shown in Figure 2.7, which constitutes a large portion of the dataset. Text reviews are converted into embeddings using the all-MiniLM-L6-v2 SentenceTransformer model, which generates vector representations for semantic meaning, making them suitable for similarity comparison and sentiment analysis. Sentiment analysis is performed using the distilbert-base-uncased-finetuned-sst-2-english model [74], classifying reviews as positive or negative with a confidence score between 0 and 1. This score is adjusted as follows:

$$\text{review\_sentiment['confidence\_score']} = \begin{cases} 1 - \text{row['confidence']} & \text{if row['sentiment'] == 0 (Negative)} \\ \text{row['confidence']} & \text{if row['sentiment'] == 1 (Positive)} \end{cases}$$

Higher confidence scores indicate stronger positive sentiment, while lower scores suggest negative sentiment. Review data is crucial because a 5-star rating scale doesn't capture the full range of sentiment, often missing emotional context or nuanced feelings [75]. Users frequently give average ratings, which reduces the discriminative power [76]. Using reviews helps the model better understand user opinions and provides richer insights for more accurate recommendations.

The heatmap analysis, as shown in Figure 2.8, shows that rating is not significantly correlated with the number of votes, comments, or review length. A moderate correlation exists between the number of votes and comments, and a weaker correlation exists between the number of votes and review length. These findings suggest that user engagement metrics and review length do not significantly impact the book rating in this dataset. Using the number of votes as a weighting factor would likely be problematic, as shown in Figure 2.3, which presents a plot showing that most reviews have no votes at all. This would result in most reviews having minimal influence on the analysis, as only a small proportion of reviews would be heavily weighted.

To improve user differentiation, a categorical feature is created by aggregating each user's top four most frequently read genres. This approach balances diversity and specificity across 12 genres, providing a meaningful representation of reading preferences. The genre distribution in Figure 6.5a demonstrates a clear distribution of genres across the dataset, justifying the selection of the top four genres to capture a representative yet manageable set of user preferences.

In recommendation systems, users and books must appear in all sets (train, test, and validation) to make sure the recommender function can be evaluated properly. A typical dataset split may lead to some books or users appearing only in the test set, which makes it impossible to generate

recommendations for them. To avoid the issue of unreliable predictions, users and books with only one interaction are removed before splitting the dataset to make sure that every book and user in the test set has appeared in the training set, leading to more reliable predictions. Data points with only one interaction are then added back into the training set. Although this slightly alters the 80/20 split, the trade-off is acceptable for a diverse and well-represented test set, improving evaluation metrics and recommender system effectiveness. This approach optimises data use for training while maintaining a useful test set for evaluation. The split is stratified by ratings to maintain a consistent distribution across both sets, ensuring a fair evaluation.

SMOTE [77] is unsuitable for recommender systems due to its assumption of a continuous feature space and synthetic interactions distort meaningful relationship. Matrix factorisation and graph-based models rely on real interactions, and synthetic data compromises their integrity. As shown in Figure 2.7, 0 ratings dominate the dataset, so adding the book or user average rating could significantly impact it.

To balance rating distributions, as shown in Figure 2.4, the most frequent rating will be used as the target, and underrepresented ratings will be duplicated. Random noise (-0.1 to 0.1) will be added to reduce overfitting, and the dataset will be shuffled to improve generalisation. A logarithmic transformation (np.log1p) will be applied to the ratings to normalise the range and improve model stability while highlighting subtle user preferences.

5.2 SVD

This model uses the complete user-book interactions, reviews, and detailed interaction datasets. To manage the computational load, the rows where the rating was missing (rating = 0) were extracted from both the reviews and interactions datasets. These rows represented users who had interacted with books but no rating was available. The User IDs associated with these missing ratings were then used to filter out the corresponding entries from the complete user-book interactions dataset, ensuring that only the relevant data was processed.

2% of the rows from the original dataset were reintroduced after filtering to add variety to the data and prevent a significant reduction in the diversity of the dataset,. This reintroduction helped to keep some of the missing rating data and made sure that the model had access to a wider range of interactions. This reduced the dataset size while also keeping good balance between computational efficiency and the model's ability to generalise.

The dataset was stratified different rating classes to make sure that each class was equally represented in the training set. The ratings were normalised as outlined earlier. To optimise the model's performance, a grid search was used with two-fold cross-validation. The best configuration achieved a mean cross-validated RMSE of approximately 0.49, indicating that the model was able to make relatively accurate predictions for missing ratings.

Once trained, the model was used to impute missing ratings from a merged dataset that included the output from NMF and GATv2Conv models. The SVD model was applied to predict ratings for the missing entries. In cases where a user or book was not present in the training set, the default behaviour of the SVD model is to predict the global mean. However, to preserve the integrity of the recommendations, this default prediction was overridden so that the missing ratings were left as null. The model successfully imputed approximately 25,000 missing ratings. These imputed ratings were rounded to the nearest whole number so that the model's output was in line with the format expected for further evaluation and use in the other models.

5.3 HDBSCAN

This model uses book feature embeddings as generated above and UMAP [78] as it can preserve both local and global structures in high-dimensional data. With 10 components UMAP balances efficiency and structural integrity to capture relationships better than PCA [79] [80]. A neighbourhood size of 300 and a minimum distance of 0.0 ensured tightly clustered points. The Cosine metric was used for as it is effective for high-dimensional text data. UMAP's low-memory mode improved scalability for the large dataset.

Embeddings were then L2-normalised and combined with HDBSCAN's Euclidean distance this approximated Cosine similarity more efficiently. HDBSCAN identified dense regions in the UMAP-reduced embeddings with hyperparameters optimised through a grid search based on Davies-Bouldin and Calinski-Harabasz Indexes (DBI, CH). Increasing the min cluster size improved the CH score but risked 20% of books being classified as outliers, the final implementation achieved 0% outlier rate.

The leaf cluster selection method identified sub-clusters by selecting leaf nodes from HDBSCAN's tree which is useful as many books span multiple themes. Instead of grouping all horror books, it can identify sub-genres like gothic horror. Soft clustering generated probabilistic membership vectors for each book to capture this thematic overlap to allow for genre-spanning suggestions. The top 5 membership vectors were stored for

recommendations and the hard cluster labels were kept for evaluation.

The HDBSCAN recommender for low interaction users generates recommendations for each of their read books individually. Each book is linked to its top clusters along with the strength of its membership, reflecting how strongly it belongs to each theme. The recommender then searches for other books that share at least one of these top clusters, provided the cluster membership probability is greater than 0.01. Candidate books are compared using Euclidean distance in the embedding space. If a book has weak or no cluster association, the function falls back on a global similarity search using FAISS, so recommendations are always provided. The results from all books are gathered, duplicates are removed, and the final list is sorted by similarity score (0-1) in descending order to prioritise the most relevant matches. This provides diverse and meaningful recommendations even with minimal user interaction.

A threshold of 0.1 was initially tested, but 80% of books didn't meet it (compared to 33% at 0.01), limiting the recommendation pool. Using 0.1 would make recommendations rare, reducing matches. The 0.01 threshold balances relevance and exploration, allowing for more diversity, especially for niche users. Soft clustering allows multiple theme memberships, providing more nuanced recommendations than hard clustering. Filtering by top clusters reduces the search space, improving efficiency, while FAISS ensures scalability. If insufficient matches are found in clusters, the system defaults to a global similarity search, ensuring recommendations. If both approaches fail, a random selection guarantees recommendations.

When a new user arrives their preferences can't be inferred so a subset of clusters is randomly selected to ensure variety and expose the user to diverse genres. A 0.01 probability threshold filters books that are strongly associated with the chosen clusters, to keep the most relevant books. The remaining books are ranked by the L2 norm of their embeddings, with those closer to the origin considered more representative of the clusters and the user's assumed preferences. Distances are converted to similarity scores using the formula: $similarity = \frac{1}{1+distance}$. If the cluster approach doesn't return enough books, the system falls back on a global similarity search using FAISS. If both methods fail, a random selection is made. This makes sure that even new users are exposed to meaningful recommendations, balancing variety, relevance, and user engagement.

5.4 GATv2Conv

The model uses Complete Book Reviews, Review Embeddings, Book Features, each user's most common genres, and Imputed Ratings (from SVD). Reviews are handled separately to avoid dataset shrinkage due to missing rating values. The dataset was split into training, validation, and test sets with an 80/10/10 ratio. Minority classes were upsampled to 75% of the majority class size to reduce computational load and ensure diversity while normalising the data. Ratings were normalised as discussed previously.

User genre preferences and book genres were one-hot encoded as node features in the GATv2Conv graph. The dataset was structured into edge index and edge attribute formats, with the edge index defining user-book connections and edge attributes including ratings, sentiment scores, and review embeddings, enabling the model to capture both user-book interactions and their characteristics.

Batch training (size = 4) was used for memory efficiency and stable gradient updates given computational constraints. Due to the small batch size, layer normalisation was preferred over batch normalisation, as it normalises across features, providing consistent outputs and improving model stability and convergence. The model consists of 5 GATv2Conv layers, each using multi-head attention to aggregate information from neighbouring nodes. The architecture was designed with 30 hidden channels and 25 attention heads to balance computational efficiency with the ability to capture complex patterns.

A dropout rate of 0.2 was used to prevent overfitting by randomly omitting features during training. The Exponential Linear Unit (ELU) activation function was selected to introduce non-linearity and improve the model's learning capacity. A learning rate of 0.00001 was chosen for stable convergence, as higher learning rates resulted in unstable training. The AdamW optimiser was used because it offers effective weight decay regularisation, eliminating the need for separate tuning of learning rate and weight decay parameters, and it converged more efficiently than SGD.

A weight decay of 1e-4 also helped to regularise the model. A learning rate scheduler (step size = 20, gamma = 0.1) was used to reduce the learning rate progressively to improve convergence. The MSE loss function was used due to its penalisation of larger errors, faster convergence and easier interpretability. Alternatives like SmoothL1 Loss were tested, but MSE converged faster and offered clearer results. Early stopping was applied (patience = 10, delta = 0.0001) to reduce overfitting and select the best model based on validation performance.

The recommender function computes the predicted rating for each unread book by taking the dot product of the user and book embeddings and denormalising it. The dot product, chosen for its speed and efficiency in large datasets, directly measures the alignment between user and book features, with higher values indicating stronger similarity. After calculating predicted ratings for all unread books, the system returns the top 5 predicted books, sorted by similarity.

5.5 NMF

The User-Book Interactions dataset was preprocessed through splitting, balancing, and normalisation, as discussed previously. To address class imbalance without significantly increasing computational complexity, the minority classes were upsampled until they reached 75% of the size of the majority class. This created a more balanced distribution of interactions, which helped improve the model's learning while keeping processing time manageable.

This model uses SVD-predicted ratings for books that users have read but not rated, helping to reduce the impact of missing data and improve the general coverage of recommendations. Hyperparameters were optimised using GridSearchCV with RMSE as the objective function, to produce accurate and generalisable predictions.

The recommender function provides book recommendations along with predicted ratings and interpretable explanations based on the model's latent factors. It starts by retrieving all unique books from the user-item interaction dataset and filters out those already rated by the target user. The NMF model then predicts ratings for each of the remaining books. These predictions are sorted in descending order to identify the most relevant books for recommendation. Since the NMF model operates on normalised rating values, the predicted ratings are denormalised to return them to their original scale. This step improves the interpretability of results, making the ratings easier to understand in terms of familiar values such as a 5-star scale.

For each recommended book, the function retrieves the user's and book's latent feature vectors, supporting explainability. It computes their element-wise product to reveal how each factor influences the predicted rating, ranking contributions by absolute value. This highlights key factors shaping the recommendation. The function returns five books, each with a predicted rating, the most influential latent features and their relative influence (e.g. Latent Feature 13: 0.67, Latent Feature 5: 0.21), enhancing both relevance

and transparency.

5.6 Hybrid Pipeline

The main function picks the most suitable recommendation strategy by first checking how many books a user has rated and how many they have reviewed. Based on these numbers, it chooses the best approach and calls any of the recommender functions detailed above, as discussed and illustrated in Figure 2.2. For high interaction the system uses a ranked weighted recommendation strategy.

Instead of simply averaging the predicted ratings from the NMF and GATv2Conv models the system combines their outputs by ranking and then weighting them. This helps to balance the strengths of each model without losing their predictive power. Both models generate their own recommendations, sorted from highest to lowest rating, and then book is assigned a rank based on its position in the list. A lower number means higher confidence from the model. Rank 1 goes to the top book, rank 2 to the second etc.

Next, the system applies a weighting factor to the ranks from each model. Since the NMF model performed better in terms of precision, it is given more influence in the final list. NMF is weighted at 0.6 and GATv2Conv at 0.4. Precision was prioritised as the goal is to return a small, highly relevant set of recommendations, making it more important to avoid irrelevant results than to capture every possible relevant one. This is so users receive fewer but more accurate suggestions, aligning with the system's aim of recommending books that align with users past preferences.

Rank-based weighting was chosen as it focuses on the relative ordering of items rather than absolute values. This makes the combination more robust to minor variations in predicted scores, especially near the top of the list, where small differences can significantly affect relevance. It also makes sure that the most confidently recommended books from each model are given priority, so users receive a concise and highly relevant set of suggestions.

For books that appear in both lists, their scores are combined using the weighted average of their ranks. The final recommendations are sorted based on these combined scores and a lower score means a stronger recommendation overall, so the final list reflects the strengths of both models and gives more weight to the one that is more accurate.

Results & Evaluation

The results of the different models and recommenders offer information on how effectively these approaches address the objectives and reveals several challenges that need refinement. This chapter presents the evaluation results for the individual and hybrid recommendation models. The chapter also examines the effectiveness of combining collaborative filtering with graph-based models and highlights potential areas for future improvement.

6.1 Model Evaluation Results

Table 6.1: HDBSCAN Clustering Metrics

Metric	**Value**
Davies-Bouldin Score	0.729
Calinski-Harabasz Score	1294.410
Silhouette Score	0.172
Avg Compactness	0.051
Avg Separation	0.141
% Outliers	0.000

HDBSCAN clustering enhances personalisation by addressing the cold-start problem for low-interaction users through the identification of dense groups of similar books.

The metrics presented in Table 6.1 are based on hard clustering, where each item belongs to a single cluster. However, the recommendation system uses the top 3 soft clusters when generating recommendations, allowing books to belong to multiple clusters. This soft clustering better captures overlapping preferences, offering a more personalised recommendation experience. Although hard clustering metrics reflect the structure of clusters they may undervalue the recommendation quality in soft clustering contexts.

The optimal cluster configuration, with a minimum cluster size of 100 and an epsilon value of 0.5, led to compact and well-separated clusters, as seen in the low Davies-Bouldin index (0.729) and the high Calinski-Harabasz Index (1294.41). The compactness score (0.051) indicates that items within clusters share strong similarities, while the moderate separation score (0.141) suggests reasonable cluster distinctions. No outliers were detected, suggesting that every item was assigned to a meaningful group, although niche items with unique themes may have been absorbed into larger clusters.

The Silhouette Score of 0.172 implies that some clusters overlap or are ambiguously defined. This may be due to the minimum cluster size of 100, which leads to larger, more generic clusters. This trade-off affects

the granularity of recommendations and may limit the system's ability to provide highly personalised suggestions. This highlights the challenge of maintaining a balance between achieving meaningful clusters and ensuring recommendations remain relevant.

HDBSCAN contributes significantly to the recommendation system by offering similarity scores for recommended books based on the embeddings. This makes the recommendations easily understandable for users and aligns with the project's goal of explainability. However, further work is needed to balance the trade-off between reliability and granularity. Refining the clustering parameters to allow outlier detection or exploring alternative clustering methods could improve the system's ability to address niche interests and provide more targeted recommendations. This could also help optimise the system's adaptability for users with diverse or emerging preferences.

The values presented in Table 6.2 were derived from the evaluation function shown in Figure 6.1. The value of k=5 was selected to balance recommendation diversity with practical usability. A smaller k makes sure that only the most relevant items are shown, promoting a higher standard of recommendation quality. It also offers a meaningful basis for evaluating ranking metrics, as users are unlikely to engage with lengthy recommendation lists in typical usage.

Table 6.2: Performance Comparison of Individual Models (Relevance Threshold >= 4)

Metric	**SVD**	**NMF**	**GATv2Conv**
RMSE	0.346	0.2590	0.2548
MAE	0.267	0.1950	0.1939
Precision@5	0.339	0.7920	0.3624
Recall@5	0.925	0.4972	0.9593
nDCG@5	0.944	0.9936	0.9757
mRR@5	0.943	0.9957	0.9751
mAP@5	0.938	0.9905	0.9732
Novelty	0.999	0.9930	0.9997
Hit Rate	0.957	1.0000	0.9812
User Coverage	0.956	1.0000	0.9812

The relevance threshold was set at 4 on a 1–5 rating scale, under the assumption that ratings of 4 and above represent a strong indication of user satisfaction. It should be noted that both k=5 and the threshold of 4 are ultimately arbitrary choices. They were determined based on conventions and practical considerations rather than strict theoretical justification. Future iterations could explore alternative values for k and relevance thresholds and adapting them dynamically to different user segments.

SVD shows strong performance in recall (0.925) but its comparatively lower precision (0.339) shows that many of these recommendations are less spe-

cific to the user's preferences, reflecting a tendency towards breadth rather than depth, which is a common limitation of traditional matrix factorisation techniques. The ranking metrics are slightly weaker compared to those of NMF and GATv2Conv, showing that SVD struggles to prioritise the most relevant items as effectively. While the high novelty score (0.999) suggests that SVD recommends less popular books, this may stem from suggesting obscure items that are not necessarily meaningful to the user.

SVD may compromise on personal relevance and ranking quality although it is not directly used in the recommender pipeline, it is beneficial to evaluate it on all the same metrics as it directly influences the performance of the other models. Some of the weaknesses observed in the NMF and GATv2Conv models could be traced back to the imputation process. This suggests that the role of SVD may need to be reconsidered by exploring alternative imputation techniques or adjusting SVD's influence within the hybrid system.

NMF has a decent precision score (0.7920) and great ranking metrics (nDCG@5: 0.9936, mAP@5: 0.9905, mRR@5: 0.9957), showing its ability to generate highly relevant recommendations that are well-ordered, improving the user experience by placing the most relevant suggestions at the top. NMF's recall (0.4972) is lower, indicating a more focused range of retrieved items, which can help avoid irrelevant suggestions. The novelty score (0.9930) shows that NMF recommends a variety of books, including some lesser known ones. This suggests that NMF effectively addresses the need for precision, relevance, and a degree of diversity, meaning a high-quality recommendation system.

The best model for GATv2Conv was loaded from epoch 21, but it may still have overfitted as seen with the plateau shown in Figure 6.7, which may explain the relatively weak Precision@5 (0.3624). While the GATv2Conv model was trained with regularisation and early stopping to prevent overfitting, there may still be some signs of suboptimal precision, possibly due to the imputation process from SVD or early stopping patience. It also suggests that while the model does well in terms of novelty and recall, it may still be less effective in fine-tuning recommendations to closely match user preferences.

GATv2Conv supports the aim of diversifying recommendations and going beyond mainstream trends with its high novelty (0.9997) and recall (0.9593). By incorporating review-based features, it likely uncovers less obvious relationships, enabling broader and less predictable suggestions. This helps reduce filter bubbles and encourages engagement by exposing users to a wider range of genres and titles. However, GATv2Conv's precision (0.3624) is much lower than that of NMF, indicating a trade-off that while the recommendations are novel and diverse, they may not always align

closely with user preferences. This lower precision may be caused by the imputation of some missing ratings from SVD, which may have affected the model more heavily as the original dataset is smaller. However, GATv2Conv still achieves strong ranking scores (nDCG@5: 0.9757, mAP@5: 0.9732, mRR@5: 0.9751).

Both NMF and GATv2Conv tend to predict higher ratings overall, as seen in Figures 6.9 and 6.8, a common issue for models trained on interaction data. This tendency to overestimate ratings may reinforce popularity effects, which the project aimed to counteract. However, the higher scores might also reflect the use of k=5 which is quite small. Since these are expected to be the most relevant suggestions, higher predicted ratings are not necessarily inaccurate, but rather indicate strong model confidence.

Figure 6.6 plots the true average ratings of the top 5 predicted books for all users using the NMF model on the test set, showing that the majority of the books fall within the 3.5 to 4.5 rating range. This suggests that the model is inclined to recommend books that are generally well-regarded by the wider user base. The concentration of ratings in this range may indicate a bias towards books that have already received positive feedback from many users, likely favouring those that are more popular or have accumulated more ratings.

This pattern could be due to popularity bias which could limit exposure to niche or lesser-known books, but as there are books outside the 3.5 to 4.5 range, with ratings as low as 2.5, shows that there is still some diversity in the recommendations. This variety is a good sign, suggesting that the system is not overly focused on a narrow set of books but can offer users a broader selection, even if it tends to prioritise more popular or higher-rated ones.

The evaluation of the individual models highlights their strengths and trade-offs. SVD's recall is good but it prioritises breadth over relevance, this could mean that there is a need for a better imputation techniques. NMF does well in precision and ranking but has lower recall. GATv2Conv's novelty and recall is strong and but its precision is weak. The soft clustering has more personalised recommendations, but these clusters may have some overlap. These models can balance the system together, but further refinement is needed for improved personalisation and niche recommendations.

6.2 Hybrid Pipeline

In the hybrid recommendation pipeline, the models are pre-trained, and so the focus shifts to inference, where the system generates recommendations based on these existing models. The goal is to recommend new, unread books, and for this task, a traditional train-test split is unsuitable. A train-test split is typically used to assess model performance on test data, but here, the models are already trained, and the aim is to predict books users haven't interacted with. Therefore, the evaluation focuses on metrics like Novelty, Hit Rate, and User Coverage, which measure how well the system introduces new and varied items, addressing issues like popularity bias in traditional recommendation systems.

Although GATv2Conv is not used on its own to generate recommendations, as its output is ranked and weighted with NMF, it is still important to evaluate it independently. This evaluation helps assess how GATv2Conv performs in isolation and how it may interact with the NMF outputs. By doing so, a more thorough comparison can be made to see whether GATv2Conv's inclusion in the hybrid model interferes with or improves the final recommendations.

One motivation for the project was to make the recommendation system explainable. The NMF model's latent factors are theoretically explainable as they represent user preference patterns. However, translating these factors into easily understandable attributes like genres or themes remains challenging. In contrast, HDBSCAN provides similarity scores based on embeddings, which are easier to interpret, as the embeddings are built using metadata like genre, author, and description.

GATv2Conv's complex attention mechanism, processing both numerical and textual data, is harder to explain. While it incorporates user reviews and features, understanding the specific reasons behind a recommendation remains complex. There were attempts to improve explainability through node embeddings and edge plots, but they turned out to be unsuitable for users. Despite the effort, they didn't provide the clarity hoped for.

The difficulty in explaining GATv2Conv highlights a key trade-off in recommendation system design. GATv2Conv improve novelty by exploring complex data relationships but they pose challenges in interpretability. This reflects the aim of balancing novelty, user satisfaction, and transparency in the recommendations. Despite these challenges, the evaluation remains aligned with the project's aim of addressing issues like popularity bias and providing recommendations that are relevant, varied, and more interpretable.

The values presented in Table 6.3 were derived from the evaluation func-

Table 6.3: Hybrid Pipeline Recommenders Performance (Relevance Threshold ≥ 4)

Metric	HDBSCAN (new users)	HDBSCAN	NMF	GAT	Ranked Weighted
nDCG@5	N/A	N/A	0.172	0.194	0.190
mAP@5	N/A	N/A	1.000	1.000	1.000
User Coverage@5	N/A	N/A	0.282	0.135	0.153
Hit Rate@5	N/A	N/A	1.000	1.000	1.000
Novelty	0.997	0.849	0.981	0.999	0.997
Item Coverage	0.299%	0.340%	0.084%	0.276%	0.223%
Diversity Ratio	77.543%	88.736%	3.417%	11.225%	9.052%

tions: Figure 6.2 for the NMF, GAT, and ranked weighted recommenders, Figure 6.3 for the HDBSCAN recommender for low-interaction users, Figure 6.4 for the HDBSCAN recommender for new users.

The results presented in the table 6.3 highlight the effectiveness of the recommender functions in addressing core challenges such as accuracy and novelty. Each model has its strengths, though they also reflect common issues in traditional systems, like popularity bias, cold-start problems, and limited novelty. As the HDBSCAN recommenders don't return predicted ratings, metrics like nDCG, mAP, and user coverage aren't applicable. It's important to note that the total number of recommended books is equal to the number of users multiplied by 5 (the number of recommendations per user). This context is useful for understanding the diversity ratio, which considers the proportion of unique recommendations. However, item coverage is based on the entire catalogue of books and reflects the extent to which the recommendations span all available books in the dataset.

It is important to consider the number of users evaluated for each model. The Hybrid model was tested on 342 users, NMF on 630 users, GAT on 302 users, and HDBSCAN models on 350 users. The variation in the number of users evaluated is crucial because it influences metrics such as User Coverage, Diversity, and the ability of each model to generalise across different user groups. Models evaluated on a larger set of users tend to perform better in terms of coverage, but they may focus more narrowly on specific user preferences, while models evaluated on smaller sets of users may offer more diverse suggestions but reach fewer users.

The random HDBSCAN recommender is designed to handle cold-start users, a key aim for this project. It achieves an impressive novelty score (0.997). However, due to the absence of user interaction data, it is not possible to evaluate relevance directly. The model clusters similar books based on content or attributes, which fits the project's goal of offering recommendations even to users with no prior history. While relevance cannot be directly measured in this case, the model ensures diversity in its recommendations (77.543%), helping to overcome the lack of personalisation in

existing systems like Goodreads.

The second HDBSCAN recommender, for users with fewer than five books read, shows a slightly lower novelty score (0.849) but higher diversity (88.736%). This highlights the model's ability to offer a broader range of recommendations despite the limited user interaction data, aligning with the project's objective to cater to users with sparse data.

The average similarity scores from the HDBSCAN-based recommendations for low-interaction users ranged from 0.97 to just below 1, with most above 0.995, as seen in Figure 6.10. This shows the system recommends highly relevant books based on users' previous preferences. Combined with a diversity ratio of 77%, the system achieves a strong balance between relevance and variety. Users receive books that feel familiar while still being exposed to new titles. This is especially useful for newer users, as it maintains personalisation without limiting discovery. The results suggest that using HDBSCAN similarity scores supports both tailored recommendations and a wide spread of items, showing the potential of this approach for adaptive, engaging suggestions.

The NMF based recommender, tested on users with sufficient reading history (630 users), shows a high novelty score (0.981) perfect mAP(1.000) and Hit Rate (1.000), suggesting that it is highly accurate for users within its scope. This aligns with the project's goal of improving recommendation accuracy. However, the model's low Diversity Ratio (3.42%) and Item Coverage (0.084%) indicate that it tends to focus on a narrow set of books, often those that are popular or highly rated. This reinforces the challenge of popularity bias and highlights the limitations in offering diverse recommendations, which the project sought to address.

The GAT based recommender, tested on 302 users, achieves the highest novelty score (0.999), underscoring its ability to recommend books outside users' past interactions. The Diversity Ratio (11.3%) and Item Coverage (0.276%) are higher compared to NMF, which shows it is better at suggesting lesser-known books and ensuring variety. However, User Coverage (13.5%) is lower, reflecting the challenges faced by graph-based models in dealing with sparsity and limited connectivity for some users, an issue known to affect the generalisability of existing recommendation systems.

The Hybrid model, which combines NMF and GATv2Conv, offers a balanced solution, bringing together the strengths of both approaches. It achieves a high novelty score (0.997) and shows improved diversity (9.05%) and item coverage (0.223%) compared to NMF alone, addressing both the need for accuracy and diversity. The nDCG score (0.190) of the hybrid model is close to that of the GATv2Conv model, indicating the successful integration of the strengths of both models. However, its User Coverage

(15.3%) remains lower than that of NMF, suggesting that it doesn't reach as many users.

All the recommender functions exhibit strong genre coverage, with all 16 genres represented in the top 5 recommendations for valid users, as seen in Figure 6.5. This indicates that the system is successfully covering a broad range of genres, providing users with varied and comprehensive recommendations. Such diversity in recommendations is crucial for improving user engagement and satisfaction, as it helps prevent recommendations from being overly focused on a narrow set of genres, which can occur if the system only caters to users' past preferences or popular trends. The inclusion of all genres in the top 5 recommendations also aligns with promoting exploration and novelty in the system.

While each recommender system includes all 16 genres, their behaviours differ when recommending less common genres. The NMF, GATv2Conv, and hybrid recommenders, see Figures 6.5b 6.5c 6.5d, tend to focus heavily on popular genres like Fiction, Fantasy, and History, leaving lesser-represented genres like Poetry and Graphic Novels recommended less frequently. The cluster-based recommenders are more effective at highlighting niche genres like Comics and Graphic Novels, showing its strength in catering to users with more specialised or unconventional tastes 6.5e 6.5f. These differences underscore how algorithmic designs influence each model's ability to diversify its recommendations and adequately represent the full spectrum of genres.

The dataset's formatting for book series information was inconsistent, which made it difficult to identify whether multiple recommendations belonged to the same series so users may receive several books from a series they had already engaged with. As a result, the diversity and item coverage metrics may have been negatively affected, since recommendations could have been unintentionally centred around closely related titles.

The results demonstrate that the models effectively address the project's motivation to improve recommendation accuracy, novelty, and explainability, while also tackling the cold-start problem. Each model excels in different areas but there is still room for improvement, particularly in achieving higher user coverage without compromising novelty. Addressing these issues could further enhance user satisfaction and engagement, while also improving recommendations for new users, and so aligning more closely with the project's original goals.

Conclusion

The hybrid approach in this project provides a diverse and relevant range of recommendations for low interaction and new users and so has effectively addressed the cold-start problem. The similarity scores from HDBSCAN showed that users received highly similar book recommendations showing that the system successfully aligns recommendations with users' read books. This focus on similarity and relevance may have come at the expense of the lower item coverage. Future work could focus on refining similarity measures to better capture user preferences and improve the relevance and diversity balance. This could involve incorporating user profile data or using reinforcement learning to adapt the model based on user interactions.

GATv2Conv's explainability and transparency are weak as the complexity of the attention weights and embedding limits its ability to offer easily interpretable explanations. Exploring alternative graph-based models or hybrid approaches that prioritise explainability could help achieve a better balance between predictive accuracy and user transparency. NMF latent features offer some explanation to users, but again their interpretability is limited, making it difficult to link these factors to features such as genres or themes.

Topic modelling could provide clearer and more comprehensible rationales for recommendations to help improve user trust and system transparency. The HDBSCAN recommender also helped the system be more explainable by using similarity scores and the recommendations can be easily traced back to the books the user has read and their features.

The evaluation was guided by a diverse set of metrics to assess both the relevance and discovery potential of the system. The novelty metric used is based on the frequency of book interactions and provided some insight into highlighting less frequently read items. The novelty scores scores the recommender functions are high showing that the system can recommended books that are less known or less popular. However, this approach does not capture the uniqueness of items across different user profiles. A more refined novelty metric that takes into account the distribution of interactions across different groups would allow for better identification of novel items, and so improve user engagement.

The range of metrics used at each stage of the system development shaped the subsequent decision-making. At first user coverage and novelty weren't used in the individual model evaluation. After they were added, it became clear that the models were already performing well in terms of discovery so

the hybrid pipeline wouldn't need much tuning to increase exploration.

In hindsight, experimenting with the inclusion threshold for the hybrid pipeline could have improved the system's evaluation. Requiring users to have read at least 15 books, rather than just 5, might have produced more stable and reliable metrics by focusing on users with richer interaction histories.

The study "Factors Influencing Readers' Interest in New Book Releases" [81] highlights the significant role of visual appeal in attracting reader interest. Incorporating additional features, such as book cover designs, could enrich the recommendation process. Using CLIP (Contrastive Language-Image Pre-training) [82] to combine visual and textual features could offer a more comprehensive understanding of reader preferences.

The current ranked-weighted hybrid approach is simplistic and could be improved with more advanced techniques. Reinforcement learning could dynamically adjust model weights based on real-time user interactions, allowing the system to optimise for either accuracy or diversity. This could help in providing a more responsive recommendation experience, particularly in cases where user preferences shift over time or in response to new content. Meta-learning techniques could determine the most suitable model for different user groups.

The experimentation with different weighting factors for each model helped to determine the most balanced approach. More exaggerated weights, such as 0.7 and 0.3, placed excessive emphasis on one model. A weighted approach was also tried but did not sufficiently incorporate both models, limiting diversity in the recommendations. These trials supported the final decision to use the ranked weighted approach to make sure that both models contributed.

The results show that the hybrid approach can significantly improve novelty and diversity but there are challenges with balancing relevance and diversity. The findings contribute to the development of more transparent, personalised, and adaptive recommendation systems that can better meet user needs. With further refinement, particularly for explainability, the system could improve user satisfaction and recommendations even more.

Appendix A

$$\text{MAE} = \frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} |r_{ui} - \hat{r}_{ui}| \tag{1.1}$$

where r_{ui} is the true rating of user u for item i, $\hat{r}_{ui}$ is the predicted rating.

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} (r_{ui} - \hat{r}_{ui})^2} \tag{1.2}$$

where r_{ui}, $\hat{r}_{ui}$ are the true and predicted ratings, respectively.

$$\text{Precision@K} = \frac{1}{K} \sum_{i=1}^{K} rel_i \tag{1.3}$$

where rel_i is the relevance indicator of the i-th item in the top K recommended items.

$$\text{Recall@K} = \frac{\sum_{i=1}^{K} rel_i}{|\text{RelevantItems}|} \tag{1.4}$$

where rel_i is the relevance indicator, and RelevantItems refers to the set of relevant items.

$$\text{DCG@K} = \sum_{i=1}^{K} \frac{2^{rel_i} - 1}{\log_2(i+1)}, \quad \text{nDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}} \tag{1.5}$$

where rel_i is the relevance score, and IDCG@K is the ideal DCG.

$$\text{mRR@K} = \frac{1}{|\mathcal{U}|} \sum_{u\in\mathcal{U}} \frac{1}{\text{rank}_u} \tag{1.6}$$

where rank_u is the rank of the first relevant item for user u, $\mathcal{U}$ is the set of users.

$$\text{mAP@K} = \frac{1}{|\mathcal{U}|} \sum_{u\in\mathcal{U}} \frac{1}{|\text{Rel}_u|} \sum_{k=1}^{K} P_u(k) \cdot rel_u(k) \tag{1.7}$$

where $P_u(k)$ is the precision at rank k, and $rel_u(k)$ is the relevance at rank k.

$$\text{HitRate@K} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \mathbb{I}(\text{Hit}_u@K) \tag{1.8}$$

where $\mathbb{I}(\text{Hit}_u@K)$ is 1 if user u has a relevant item in the top K, otherwise 0.

$$\text{UserCoverage} = \frac{|\{u \in \mathcal{U} : R_u \cap G_u \neq \emptyset\}|}{|\mathcal{U}|} \tag{1.9}$$

where R_u is the set of recommended items for user u, and G_u is the set of ground-truth relevant items for u.

$$\text{ItemCoverage} = \frac{|\bigcup_{u \in \mathcal{U}} R_u|}{|\mathcal{I}|} \tag{1.10}$$

where $\mathcal{I}$ is the set of all items, and R_u is the set of recommended items for user u.

$$\text{Diversity} = 1 - \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|R_u|(|R_u| - 1)} \sum_{\substack{i,j \in R_u \\ i \neq j}} \text{sim}(i, j) \tag{1.11}$$

where $\text{sim}(i, j)$ is the similarity between items i and j.

$$\text{Novelty@K} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \sum_{i \in R_u@K} -\log_2 \left(\frac{pop_i}{|\mathcal{U}|} \right) \tag{1.12}$$

where pop_i is the popularity of item i, and $R_u@K$ is the set of top K recommended items for user u.

$$\text{cosine}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} \tag{1.13}$$

where $\mathbf{a}$ and $\mathbf{b}$ are two vectors, and $\|\mathbf{a}\|$ is the norm of vector $\mathbf{a}$.

$$\text{dot}(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} a_i b_i \tag{1.14}$$

where a_i and b_i are the components of vectors $\mathbf{a}$ and $\mathbf{b}$.

$$\text{euclidean}(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{1.15}$$

where a_i and b_i are the components of vectors $\mathbf{a}$ and $\mathbf{b}$.

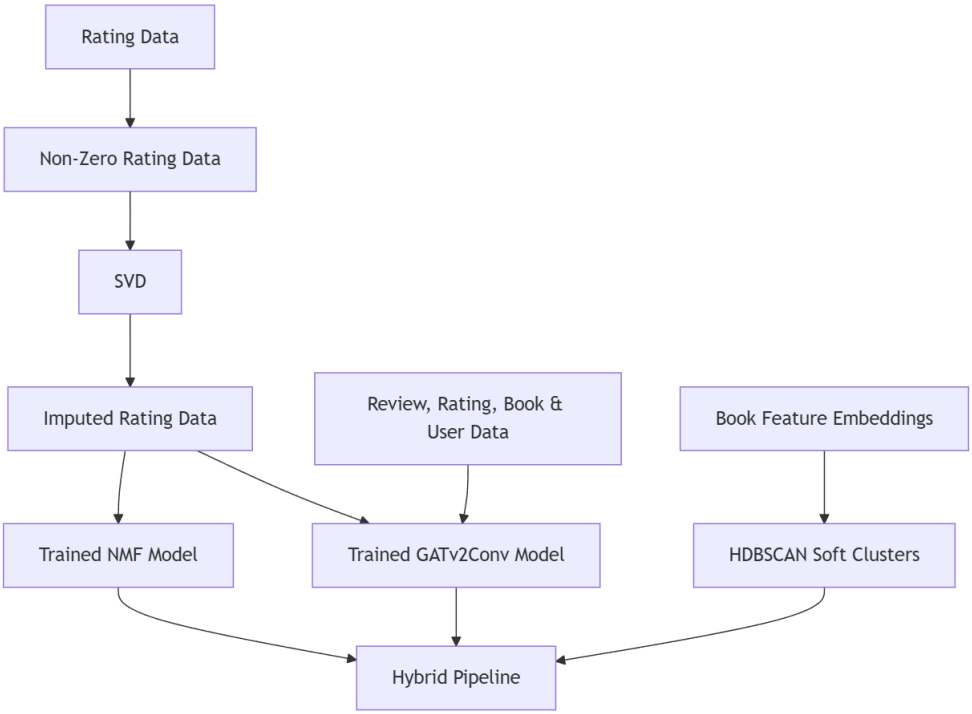


Figure 2.1: Flowchart of Model Training Setup

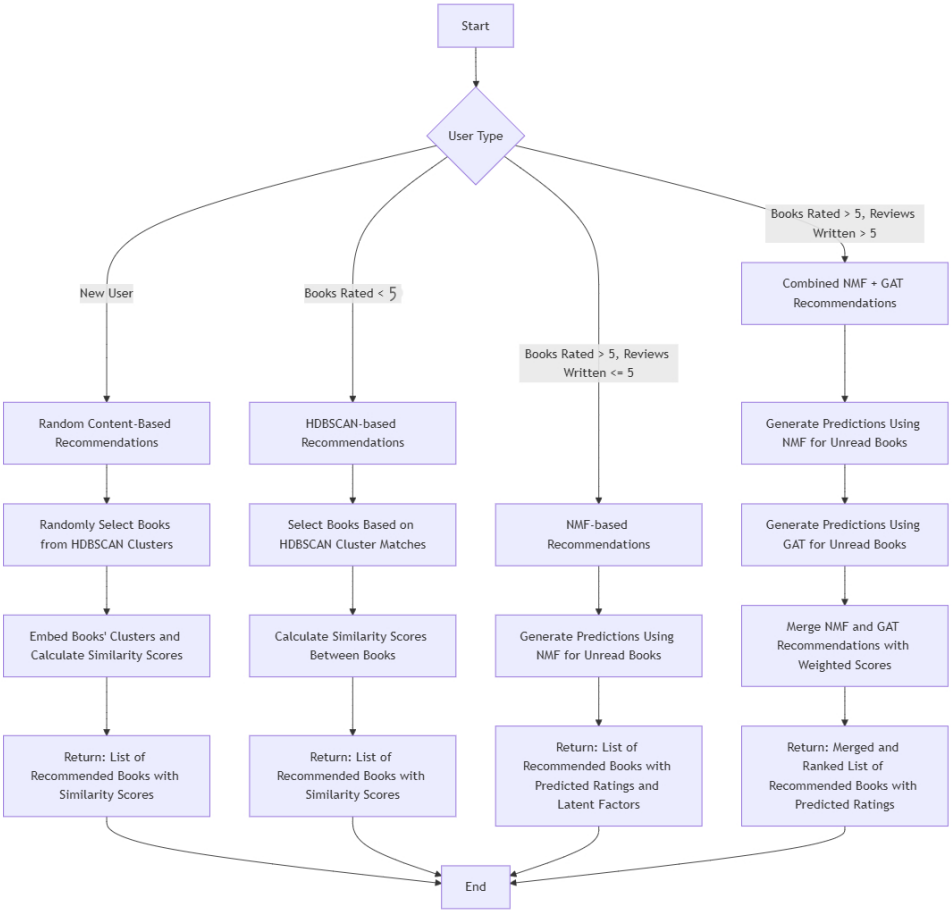


Figure 2.2: Flowchart of Hybrid Pipeline Setup

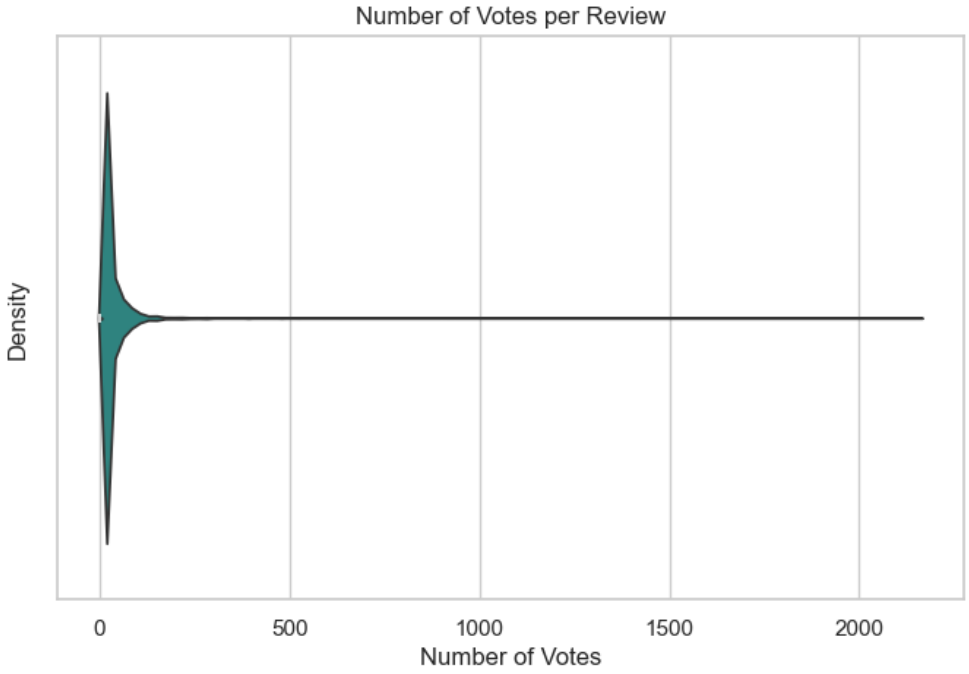


Figure 2.3: Distribution of Number of Votes Per Review

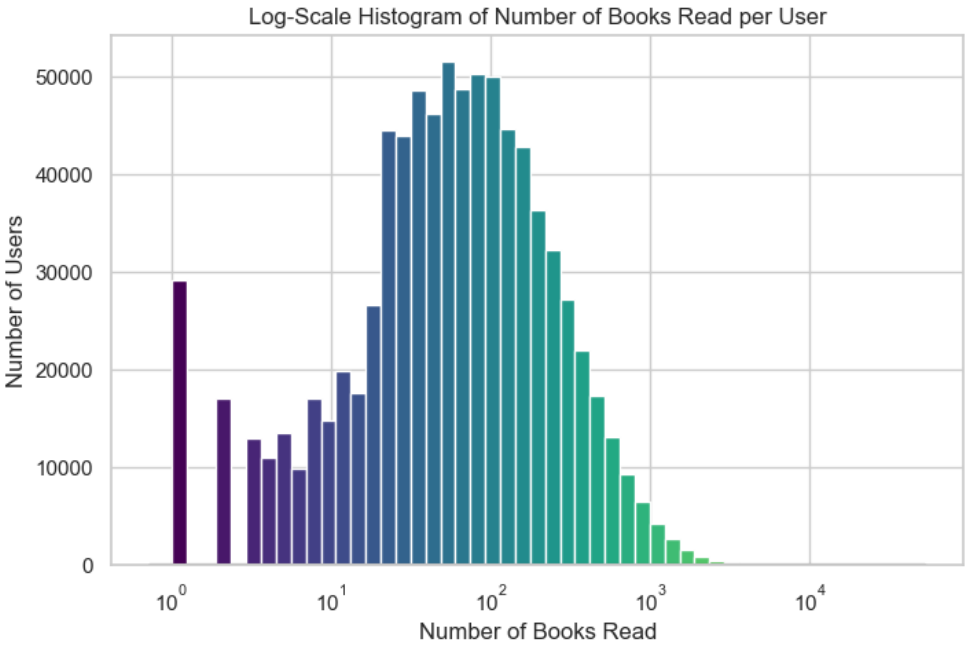


Figure 2.4: Distribution of Number of Books Read Per User

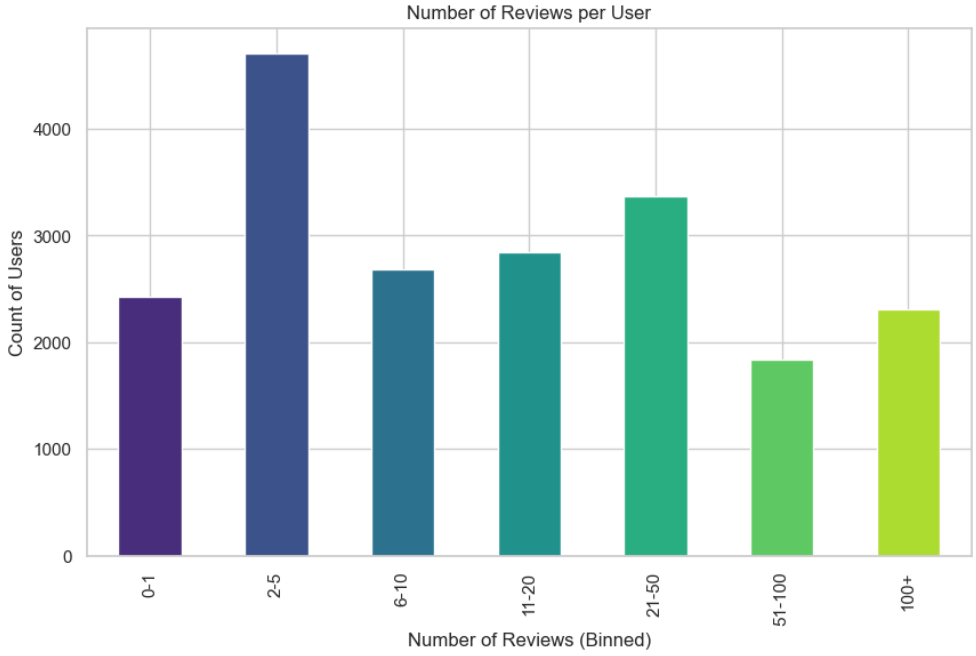


Figure 2.5: Distribution of Number of Reviews Per User

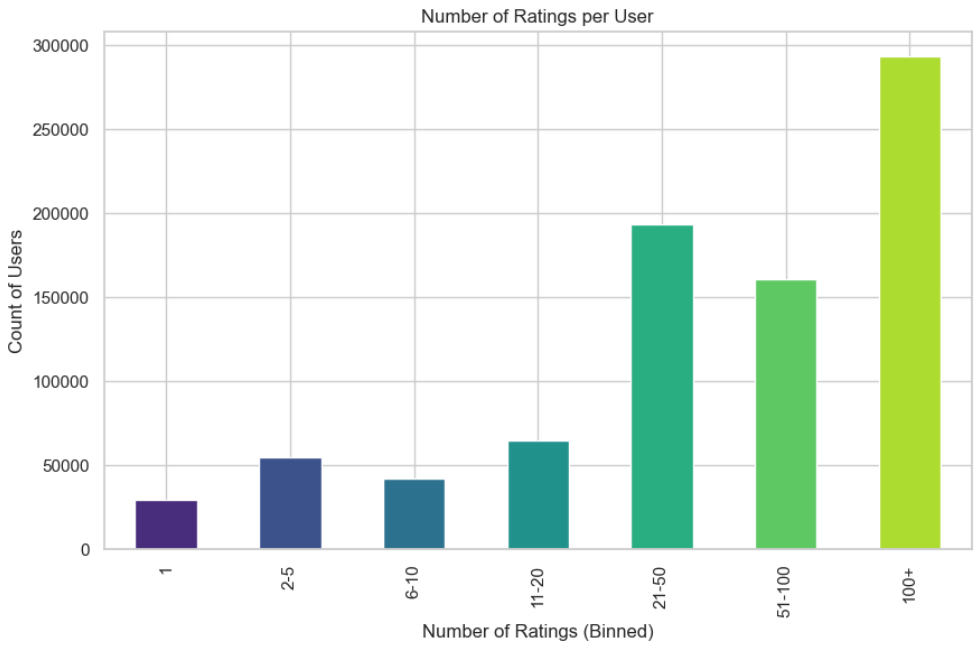


Figure 2.6: Distribution of Number of Ratings Per User

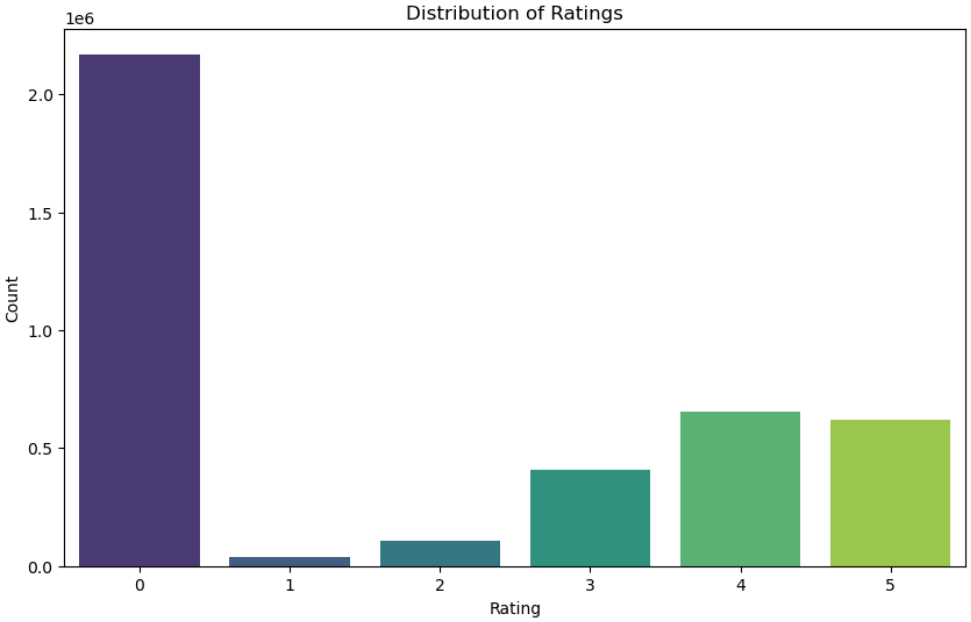


Figure 2.7: Distribution of Rating Classes

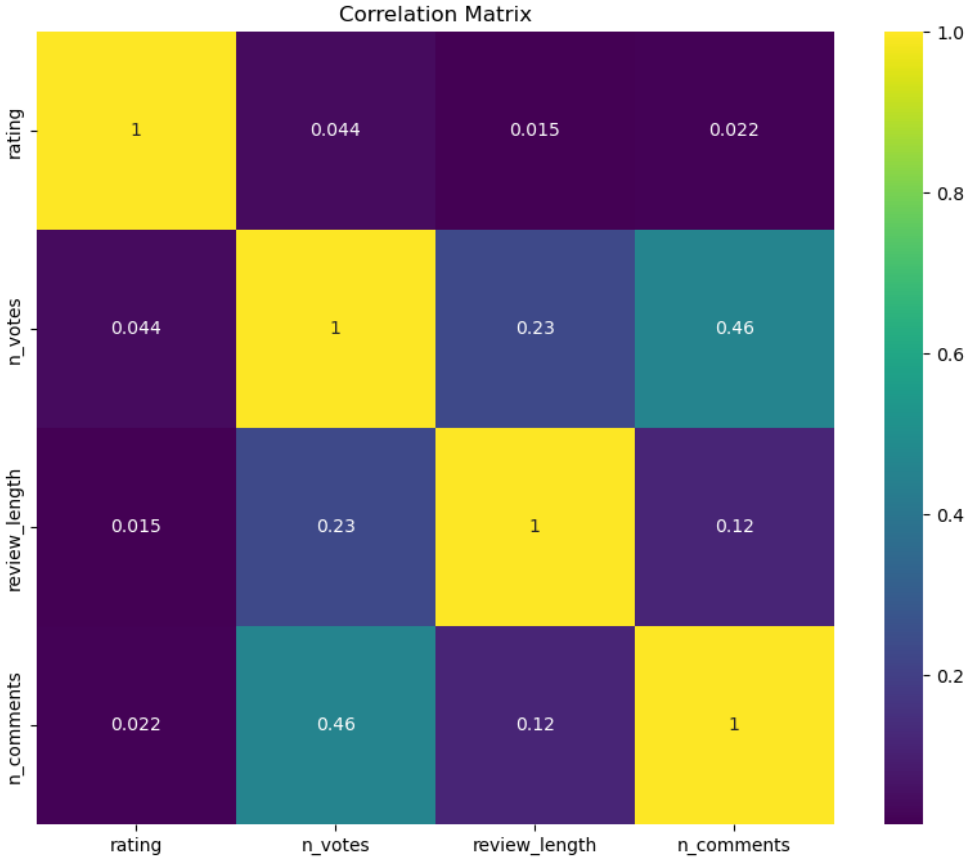


Figure 2.8: Correlation Matrix of Book Features

```python
def evaluate_model(predictions, min_rating, k, threshold, item_popularity, num_users):
    denormalized_predictions = [
        (uid, iid, true_r * (5 - min_rating) + min_rating, est * (5 - min_rating) + min_rating, details)
        for (uid, iid, true_r, est, details) in predictions
    ]

    user_est_true = defaultdict(list)
    for uid, iid, true_r, est, _ in denormalized_predictions:
        user_est_true[uid].append((iid, est, true_r))

    precisions = []
    recalls = []
    ndcgs = []
    map_scores = []
    mrr_scores = []
    novelty_scores = []
    hit_rate_scores = []
    all_recommended_items = set()
    recommended_relevant_items = set()

    relevant_items = set(iid for _, iid, true_r, _, _ in denormalized_predictions if true_r >= threshold)

    for user_id, user_ratings in user_est_true.items():
        user_ratings_sorted = sorted(user_ratings, key=lambda x: x[1], reverse=True)
        top_k = user_ratings_sorted[:k]

        top_k_items = [iid for iid, _, _ in top_k]
        all_recommended_items.update(top_k_items)
        n_rel = sum((true_r >= threshold) for (_, _, true_r) in user_ratings)
        n_rec_k = sum((true_r >= threshold) for (_, _, true_r) in top_k)
        n_rel_and_rec_k = sum((true_r >= threshold) for (_, _, true_r) in top_k)

        precision = n_rel_and_rec_k / k if k != 0 else 0
        recall = n_rel_and_rec_k / n_rel if n_rel != 0 else 0
        precisions.append(precision)
        recalls.append(recall)
        dcg = sum(
            (true_r >= threshold) / np.log2(idx + 2)
            for idx, (_, _, true_r) in enumerate(top_k)
        )
        idcg = sum(
            1.0 / np.log2(idx + 2) for idx in range(min(n_rel, k))
        )
        ndcg = dcg / idcg if idcg != 0 else 0
        ndcgs.append(ndcg)
        hits = 0
        sum_precisions = 0
        for idx, (_, _, true_r) in enumerate(top_k):
            if true_r >= threshold:
                hits += 1
                sum_precisions += hits / (idx + 1)
        ap = sum_precisions / min(n_rel, k) if n_rel != 0 else 0
        map_scores.append(ap)
        first_relevant_rank = None
        for idx, (_, _, true_r) in enumerate(top_k):
            if true_r >= threshold:
                first_relevant_rank = idx + 1
                break

        if first_relevant_rank is not None:
            mrr_scores.append(1 / first_relevant_rank)
        else:
            mrr_scores.append(0)
        novelty = np.mean([1 - (item_popularity.get(iid, 1) / num_users) for iid in top_k_items])
        novelty_scores.append(novelty)
        hit_rate = 1 if n_rel_and_rec_k > 0 else 0
        hit_rate_scores.append(hit_rate)
        recommended_relevant_items.update([iid for iid, est, true_r in top_k if true_r >= threshold])

    user_coverage = sum(1 for user_ratings in user_est_true.values() if
                        any(true_r >= threshold for _, _, true_r in user_ratings)) / len(user_est_true)
    return (
        np.mean(precisions),
        np.mean(recalls),
        np.mean(ndcgs),
        np.mean(map_scores),
        np.mean(mrr_scores),
        np.mean(novelty_scores),
        np.mean(hit_rate_scores),
        user_coverage
    )
```

Figure 6.1: Evaluation Function for NMF, GAT, SVD Test Sets

```python
def evaluate_recommendations(df_recs_pred, k, item_popularity, num_users, books, threshold=4, top_n=5):
    ndcg_list = []
    map_list = []
    novelty_list = []
    hit_rate_list = []
    users_with_recs = 0
    recommended_book_ids_all_users = set()
    relevant_users = 0

    for user_id, group in df_recs_pred.groupby('user_id'):
        top_recommended_books = group.drop_duplicates(subset=['book_id']).head(top_n)
        recommended_book_ids = top_recommended_books['book_id'].tolist()
        recommended_book_ids_all_users.update(recommended_book_ids)

        dcg = sum((1 if row['predicted_rating'] >= threshold else 0) / np.log2(idx + 2) for idx, row in top_recommended_books.iterrows())
        idcg = sum(1 / np.log2(idx + 2) for idx in range(min(len(top_recommended_books), k)))
        ndcg = dcg / idcg if idcg != 0 else 0
        ndcg_list.append(ndcg)

        sum_precisions = 0
        for idx, row in top_recommended_books.iterrows():
            if row['predicted_rating'] >= threshold:
                sum_precisions += (idx + 1) / (idx + 1)
        map_score = sum_precisions / min(len(top_recommended_books), k) if len(top_recommended_books) > 0 else 0
        map_list.append(map_score)

        novelty = np.mean([1 - (item_popularity.get(book, 1) / num_users) for book in recommended_book_ids])
        novelty_list.append(novelty)

        hit_rate = 1 if any(row['predicted_rating'] >= threshold for idx, row in top_recommended_books.iterrows()) else 0
        hit_rate_list.append(hit_rate)

        if any(row['predicted_rating'] >= threshold for idx, row in top_recommended_books.iterrows()):
            users_with_recs += 1
            relevant_users += 1

    total_books_in_catalog = books['book_id'].nunique()
    unique_recommended_books = len(recommended_book_ids_all_users)
    item_coverage = (unique_recommended_books / total_books_in_catalog) * 100
    diversity_ratio = (unique_recommended_books / (num_users * top_n)) * 100

    user_coverage = relevant_users / num_users if num_users > 0 else 0

    evaluation_metrics = {
        'ndcg': np.mean(ndcg_list),
        'map': np.mean(map_list),
        'novelty': np.mean(novelty_list),
        'hit_rate': np.mean(hit_rate_list),
        'user_coverage': user_coverage,
        'item_coverage': f'{item_coverage}%',
        'diversity_ratio': f'{diversity_ratio:}%'
    }

    return evaluation_metrics
```

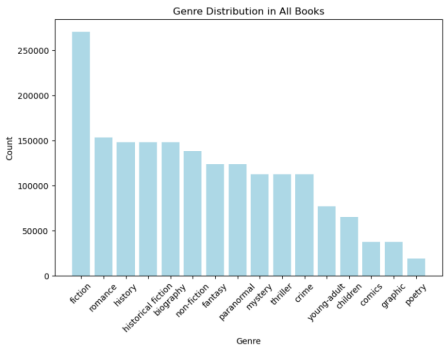
Figure 6.2: Evaluation for Weighted Ranked, NMF, and GAT Recommenders

```python
def evaluate_recommendations(recommendations_list, books, interactions, recs_per_user=5):
    flat_recommendations = []
    for user_idx, user_recs in enumerate(recommendations_list):
        user_id = f"new_user_{user_idx + 1}"
        for rec in user_recs:
            flat_recommendations.append({
                "user_id": user_id,
                "book_id": rec['book_id']
            })

    df_recs_pred = pd.DataFrame(flat_recommendations)

    recommended_book_ids = set(df_recs_pred['book_id'])
    total_books = books['book_id'].nunique()
    unique_recommended_books = len(recommended_book_ids)
    item_coverage = (unique_recommended_books / total_books) * 100

    num_users = len(recommendations_list)
    max_possible_unique_recs = num_users * recs_per_user
    print(f"\nUsers evaluated: {num_users}")
    print(f"Max possible unique recommended books: {max_possible_unique_recs}")
    print(f"Unique recommended books: {unique_recommended_books}")
    print(f"Item Coverage: {item_coverage}% ({unique_recommended_books}/{total_books} books recommended)")

    diversity_ratio = (unique_recommended_books / max_possible_unique_recs) * 100
    print(f"Diversity Ratio: {diversity_ratio}% of max possible unique recs")

    avg_popularity = None
    novelty = None
    if interactions is not None:
        book_popularity = interactions['book_id'].value_counts()
        pop_scores = [book_popularity.get(book_id, 0) for book_id in recommended_book_ids]

        if pop_scores:
            avg_popularity = sum(pop_scores) / len(pop_scores)
            print(f"Average Popularity (lower = more novel): {avg_popularity} interactions per book")

            total_interactions = interactions['book_id'].count()
            novelty = 1 - (sum(pop_scores) / total_interactions) if total_interactions > 0 else 0
            print(f"Novelty (higher = more novel): {novelty}")
        else:
            print("No popularity data available for recommended books.")

    metrics = {
        "Item Coverage (%)": (item_coverage, 2),
        "Unique Recommended Books": unique_recommended_books,
        "Total Books in Catalog": total_books,
        "Users Evaluated": num_users,
        "Max Possible Unique Recommendations": max_possible_unique_recs,
        "Diversity Ratio (%)": (diversity_ratio, 2)
    }

    if avg_popularity is not None:
        metrics["Average Popularity"] = round(avg_popularity, 2)
    if novelty is not None:
        metrics["Novelty"] = round(novelty, 2)

    return metrics
```

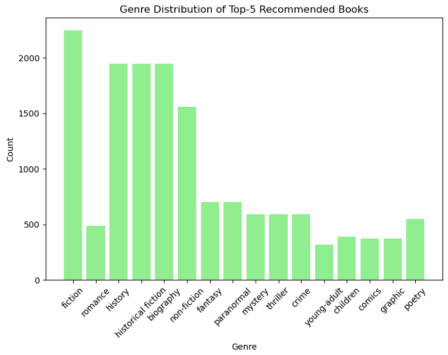
Figure 6.3: Evaluation for HDBSCAN Low Interactions Users Recommender

```python
def generate_recs_hdbscan_new_users(clustered_books, faiss_index, books, interactions,
                                    nmf_model, all_embeddings, user_id_to_index_gat,
                                    book_id_to_index_gat, reviews, top_n,
                                    sample_size, gat_weight, nmf_weight):

    max_user_id = interactions['user_id'].max() if not interactions.empty else 0
    next_user_id = max_user_id + 1

    recommendations = []
    for i in range(sample_size):
        user_id = f"new_user_{next_user_id + i}"

        user_recommendations = recommend_for_user(
            user_id=user_id,
            interactions=interactions,
            reviews=reviews,
            faiss_index=faiss_index,
            book_id_to_index=book_id_to_index,
            clustered_books=clustered_books,
            nmf_model=nmf_model,
            all_embeddings=all_embeddings,
            user_id_to_index_gat=user_id_to_index_gat,
            book_id_to_index_gat=book_id_to_index_gat,
            books=books,
            min_rating=1,
            top_n=top_n,
            final_size=top_n,
            gat_weight=gat_weight,
            nmf_weight=nmf_weight
        )

        for rec in user_recommendations:
            rec['user_id'] = user_id
        recommendations.append(user_recommendations)
        print(f"Recommendations generated for {user_id}")

    flattened_recommendations = [rec for user_recs in recommendations for rec in user_recs]
    df_recommendations = pd.DataFrame(flattened_recommendations)

    print(f"Total new users processed: {sample_size}")
    print(f"Total recommendations generated: {len(flattened_recommendations)}")

    return df_recommendations
```

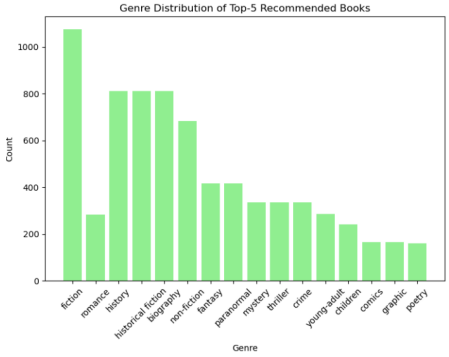
Figure 6.4: Evaluation for HDBSCAN New Users Recommender



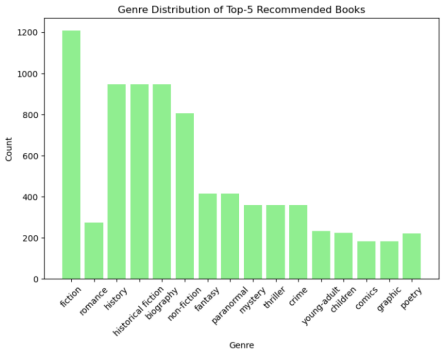
(a) Genre Count Over All Books



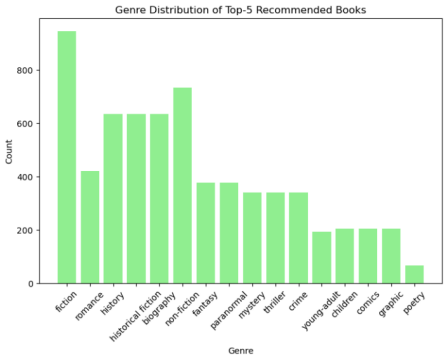
(b) Top 5 Recommendations: NMF



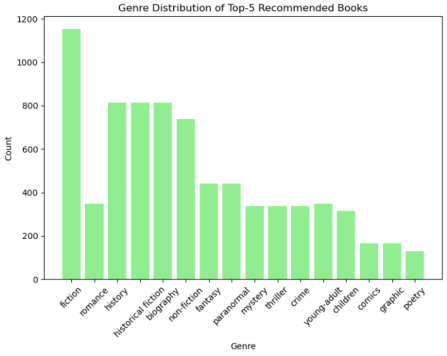
(c) Top 5 Recommendations: GAT



(d) Top 5 Recommendations: Weighted Ranked Hybrid



(e) Top 5 Recommendations: HDBSCAN for New Users



(f) Top 5 Recommendations: HDBSCAN

Figure 6.5: Genre Counts Comparison Across Recommender Models

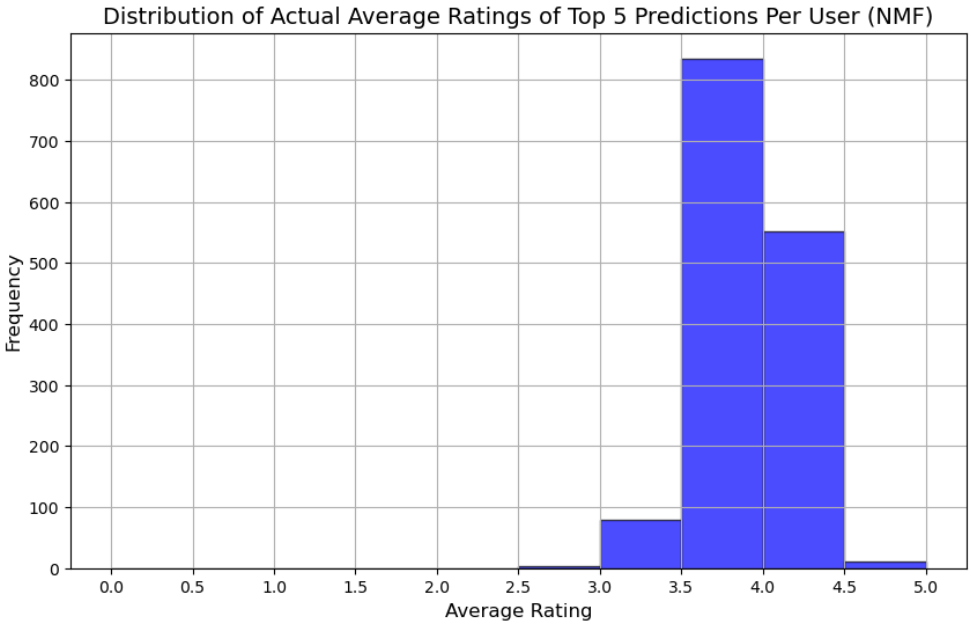


Figure 6.6: True Average Rating of Top 5 Predictions Per User – NMF Model

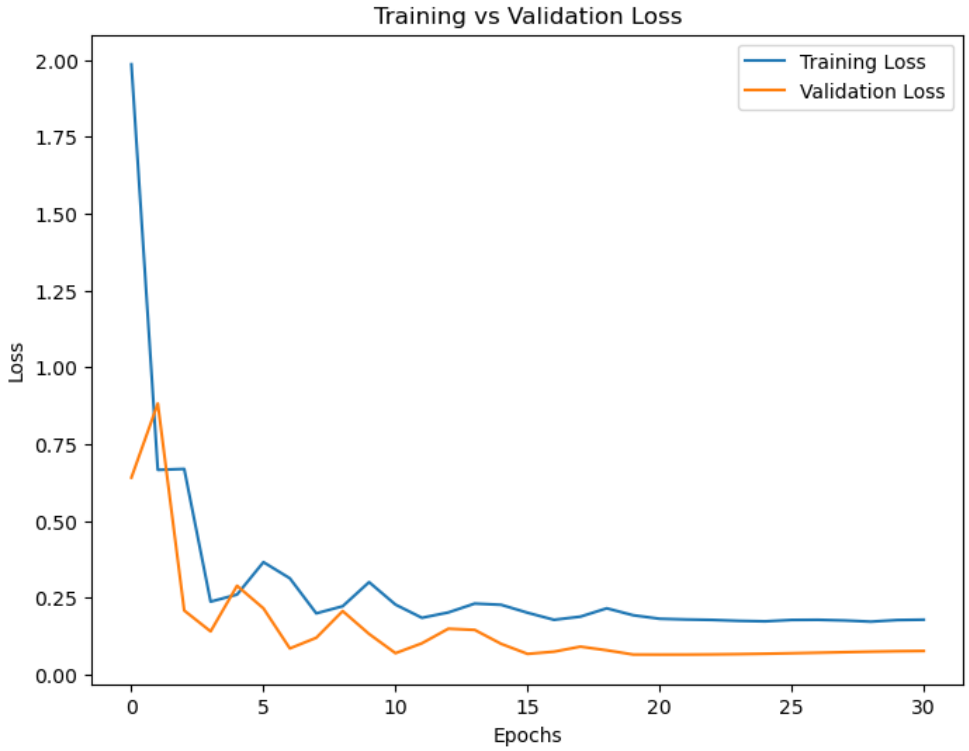


Figure 6.7: GATv2Conv Training and Validation Loss

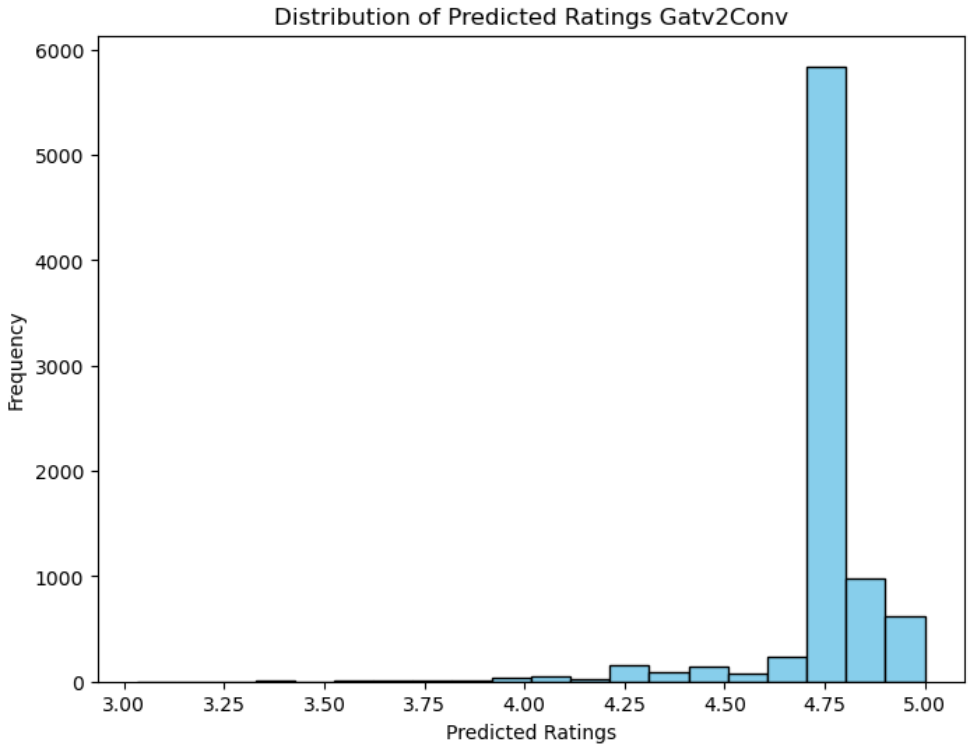


Figure 6.8: Plot of Predicted Ratings from GAT Model on Test Set

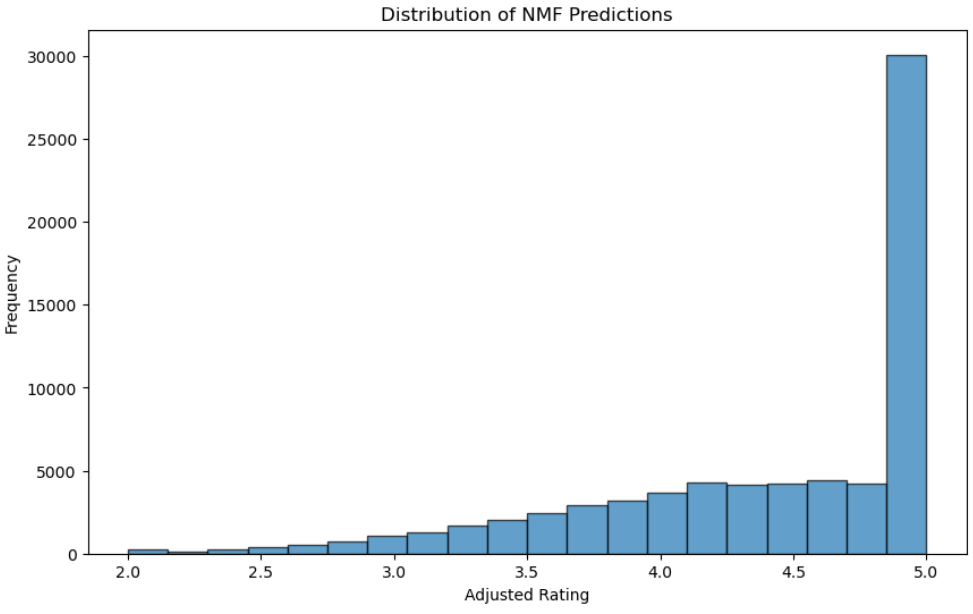


Figure 6.9: Plot of Predicted Ratings from NMF Model on Test Set

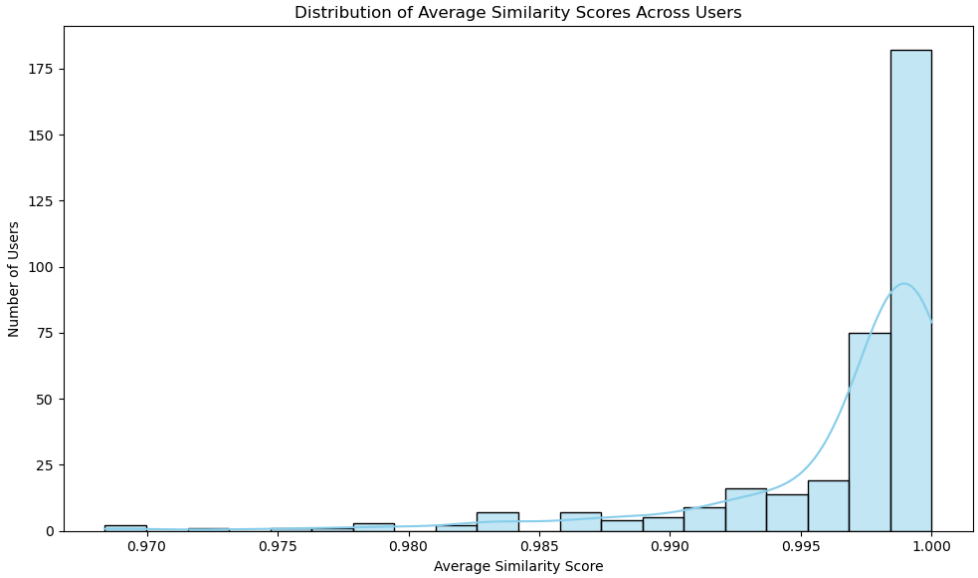


Figure 6.10: Distribution of Average Similarity Score over Users Recommendations Using HDBSCAN Recommender for Low Interaction Users (number of read books < 5)

Bibliography

[1] J. McAuley, *Goodreads datasets*, https://cseweb.ucsd.edu/~jmcauley/datasets/goodreads.html, [Accessed 09-04-2025], 2018.

[2] M. J. Pazzani and D. Billsus, 'Content-based recommendation systems,' in *The Adaptive Web*, Accessed: 2025-02-01, Springer, 2020, pp. 325–341. DOI: 10.1007/978-3-540-72079-9_10. [Online]. Available: https://link.springer.com/content/pdf/10.1007/978-3-540-72079-9_10?pdf=chapter%20toc.

[3] C. Pinela. 'Content-based recommender systems.' Accessed: 2025-02-011. (2020), [Online]. Available: https://medium.com/@cfpinela/content-based-recommender-systems-a68c2aee2235.

[4] A. Al and S. S. Hussain. 'An improved recommender system solution to mitigate the over-specialization problem using genetic algorithms.' Accessed: 2025-04-01. (2022), [Online]. Available: https://www.researchgate.net/publication/357809113_An_Improved_Recommender_System_Solution_to_Mitigate_the_Over-Specialization_Problem_Using_Genetic_Algorithms.

[5] U. Kużelewska and E. Guziejko, *A recommender system based on content clustering used to propose forum articles*, https://www.researchgate.net/publication/286493312_A_Recommender_System_Based_on_Content_Clustering_Used_to_Propose_Forum_Articles, Accessed: 2025-03-21.

[6] U. Kużelewska and E. Guziejko, *Speeding up tensor-based recommenders with clustered tag space and improving quality of recommendations with non-negative tensor factorization*, https://vbn.aau.dk/ws/files/52685302/Master_thesis_f11d625a.pdf, Accessed: 2024-12-2.

[7] D. Shukla and C. R. Chowdary, 'A model to address the cold-start in peer recommendation by using k-means clustering and sentence embedding,' *J. Comput. Sci.*, 2024, Accessed: 2024-12-17.

[8] C. contributors, *Dbscan clustering documentation*, Accessed: 2024-11-19, 2025. [Online]. Available: https://clusteringjl.readthedocs.io/en/latest/dbscan.html.

[9] L. McInnes, J. Healy and J. Melville, *How hdbscan works*, https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html, Accessed: 2024-11-11, 2025.

[10] F. Ricci, L. Rokach and B. Shapira, *Recommender Systems: Challenges and Research Opportunities*. Springer, 2011. DOI: 10.1007/978-1-4471-4496-7.

[11] B. Sarwar, G. Karypis, J. A. Konstan and J. Riedl, 'Item-based collaborative filtering recommendation algorithms,' in *Proceedings of the 10th International Conference on World Wide Web*, ACM, 2001, pp. 285–295.

[12] X. Su and T. M. Khoshgoftaar, 'A survey of collaborative filtering techniques,' *Advances in Artificial Intelligence*, vol. 2012, pp. 1–19, 2012.

[13] *Matrix factorization-based algorithms — surprise 1 documentation*, Accessed: 2025-03-18. [Online]. Available: https://surprise.readthedocs.io/en/stable/matrix_factorization.html.

[14] S. Gower, *Netflix prize and svd*, Accessed: 2025-03-13. [Online]. Available: http://buzzard.ups.edu/courses/2014spring/420projects/math420-UPS-spring-2014-gower-netflix-SVD.pdf.

[15] R. Harshman, *Singular Value Decomposition: A Tool for Understanding and Processing Data*. Springer, 1993, Chapter 6, SVD and Recommender Systems.

[16] *Nonnegative matrix factorization - an overview — sciencedirect topics*, Accessed: 2025-03-28. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/nonnegative-matrix-factorization#:~:text=Nonnegative%20matrix%20factorization%20(NMF)%20is,NMF%20only%20creates%20positive%20factors..

[17] R. Gemulla, P. J. Haas, E. Nijkamp and Y. Sismanis, 'Large-scale matrix factorization with distributed stochastic gradient descent,' in *Proc. 17th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, San Diego, CA, USA: ACM, 2011. DOI: 10.1145/2020408.2020426.

[18] Y. LeCun, Y. Bengio and G. Hinton, *Deep learning*, Accessed: 2025-03-17. [Online]. Available: https://www.nature.com/articles/nature14539#citeas.

[19] *Torch_geometric.nn.conv.gatv2conv — pytorch-geometric documentation*, Accessed: 2025-03-17. [Online]. Available: https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.conv.GATv2Conv.html.

[20] *Torch_geometric.nn.conv.gatconv — pytorch-geometric documentation*, Accessed: 2025-03-28. [Online]. Available: https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.conv.GATConv.html.

[21] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, 'Graph attention networks,' in *International Conference on Learning Representations (ICLR)*, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ.

[22] T. Kipf and M. Welling, 'Semi-supervised classification with graph convolutional networks,' *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[23] R.Selvaraju *et al.*, 'Grad-cam: Visual explanations from deep networks via gradient-based localization,' in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, IEEE, 2017, pp. 618–626.

[24] M. T. Ribeiro, S. Singh and C. Guestrin, 'Why should i trust you? explaining the predictions of any classifier,' *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016.

[25] S. M. Lundberg and S. .-. Lee, 'A unified approach to interpreting model predictions,' *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, pp. 4765–4774, 2017.

[26] S. S. et al., *Weighted hybrid technique for recommender system*, Accessed: 2025-03-18. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/930/1/012050/pdf.

[27] R. Burke, 'Hybrid recommender systems: Survey and experiments,' in *User Modeling and User-Adapted Interaction*, vol. 12, Springer, 2002, pp. 331–370.

[28] H. Ferreira *et al.*, 'A hybrid recommender system for personalized learning activities: A case study in the context of learning styles,' *Computers in Human Behavior*, vol. 109, p. 105 701, 2020.

[29] *User modeling for adaptive news access - user modeling and user-adapted interaction — link.springer.com*, Accessed: 2025-03-18. [Online]. Available: https://link.springer.com/article/10.1023/A:1026501525781.

[30] Z. ul Abideen, *One-stop guide for production recommendation systems*, Accessed: 2025-02-25. [Online]. Available: https://medium.com/@zaiinn440/one-stop-guide-for-production-recommendation-systems-9491f68d92e3.

[31] C. C. Aggarwal, *Data Mining: The Textbook*. Springer, 2015.

[32] *Faiss — ai.meta.com*, Accessed: 2025-01-5. [Online]. Available: https://ai.meta.com/tools/faiss/.

[33] Caboom.ai, *Speeding up similarity search in recommender systems using faiss*, Accessed: 2025-03-18. [Online]. Available: https://ai.plainenglish.io/speeding-up-similarity-search-in-recommender-systems-using-faiss-basics-part-i-ec1b5e92c92d.

[34] *Dot product - formula*, Accessed: 2025-01-06. [Online]. Available: https://www.cuemath.com/algebra/dot-product/.

[35] S. Rendle, C. Freudenthaler, Z. Gantner and L. Schmidt-Thieme, 'Evaluation of recommendation systems based on mean absolute error (mae),' *Proceedings of the International Conference on Data Mining (ICDM)*, pp. 387–396, 2010. DOI: 10.1109/ICDM.2010.42.

[36] Z. Ye, F. Yang, Y. Liu and Z. Yao, 'Evaluation of recommender systems,' *International Journal of Computational Intelligence and Applications*, vol. 10, no. 4, pp. 355–372, 2011. DOI: 10.1142/S1469026811002550.

[37] T. J. N. Srebro and S. Shalev-Shwartz, 'Collaborative filtering and the missing at random assumption,' *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 200–207, 2004. [Online]. Available: http://www.jmlr.org/proceedings/papers/v2/srebro04a/srebro04a.pdf.

[38] H. Steck, *Evaluation of recommendations: Rating-prediction and ranking*, Accessed: 2024-12-20. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2507157.2507160.

[39] *How to choose a threshold for an evaluation metric for large language models*, Accessed: 2024-12-20. [Online]. Available: https://arxiv.org/html/2412.12148v1.

[40] M.Bhattacharyya, *Metrics of recommender systems*, Accessed: 2025-01-22. [Online]. Available: https://medium.com/data-science/metrics-of-recommender-systems-cde64042127a.

[41] 'Evaluating and selecting recommender systems: A comprehensive approach,' *ACM Digital Library*, 2024. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3460231.3474234.

[42] L. Zhang, 'The definition of novelty in recommendation system,' *Journal of Engineering Science and Technology Review*, vol. 6, no. 3, pp. 141–145, 2013. DOI: 10.25103/jestr.063.25. [Online]. Available: https://www.researchgate.net/publication/288787908_The_Definition_of_Novelty_in_Recommendation_System.

[43] P. Vargas and F. Castells, 'Novelty-aware ranking for recommender systems,' *Proceedings of the 5th ACM Conference on Recommender Systems*, pp. 147–154, 2011. DOI: 10.1145/2043932.2043962.

[44] C. X. X. Wang Y. Guo, 'Recommendation algorithms for optimizing hit rate, user satisfaction, and website revenue,' in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, Accessed: 2025-04-15, IJCAI Organization, 2015, pp. 1820–1826. DOI: 10.5555/2832415.2832502. [Online]. Available: https://dl.acm.org/doi/10.5555/2832415.2832502.

[45] D. J. M. Ge C. Delgado-Battenfeld, 'Beyond accuracy: Evaluating recommender systems by coverage and serendipity,' *Proceedings of the 2020 International Conference on Recommender Systems (RecSys)*, vol. 12, pp. 1–8, 2020. DOI: 10.1145/3383313.3383331. [Online]. Available: https://dl.acm.org/doi/10.1145/3383313.3383331.

[46] M. H. I. Koutsopoulos, 'Efficient and fair item coverage in recommender systems,' in *Proceedings of the IEEE International Conference on Big Data Intelligence and Computing (DataCom)*, 2018, pp. 912–918. DOI: 10.1109/DASC/PICOM/DATACOM/CYBERSCITEC.2018.000-9. [Online]. Available: https://doi.org/10.1109/DASC/PICOM/DATACOM/CYBERSCITEC.2018.000-9.

[47] S. A.-Y. C. Yu L. Lakshmanan, 'It takes variety to make a world: Diversification in recommender systems,' in *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys)*, ACM, 2014, pp. 113–120. DOI: 10.1145/2645710.2645748.

[48] *Silhouette score scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html, [Accessed: 2025-01-22].

[49] *Calinski harabasz score scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.calinski_harabasz_score.html, [Accessed: 2025-01-23].

[50] *Davies bouldin score scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html, [Accessed: 2025-01-22].

[51] D. M. Blei, A. Y. Ng and M. I. Jordan, 'Latent dirichlet allocation,' *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[52] G. Adomavicius and A. Tuzhilin, 'Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,' *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005. DOI: 10.1109/TKDE.2005.99.

[53] Z. Sheng and G. Chen, 'Trustworthy recommender systems: A survey,' *Information Fusion*, vol. 63, pp. 120–135, 2021. DOI: 10.1016/j.inffus.2020.07.006.

[54] W. Hua, Y. Ge, S. Xu, J. Ji, Z. Li and Y. Zhang, 'Up5: Unbiased foundation model for fairness-aware recommendation,' in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, St. Julian's, Malta: Association for Computational Linguistics, 2024. DOI: 10.48550/arXiv.2305.12090.

[55] J. Zhang, K. Bao, Y. Zhang, W. Wang, F. Feng and X. He, 'Is chatgpt fair for recommendation? evaluating fairness in large language model recommendation,' in *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys)*, ACM, 2023. DOI: 10.1145/3604915.3608860.

[56] J. Hou, X. Wang and W. H. Wang, 'Providing item-side individual fairness for deep recommender systems,' in *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, ACM, 2022. DOI: 10.1145/3531146.3533079.

[57] R. Binns, 'Fairness in recommender systems: A sociotechnical perspective,' in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ACM, 2018, pp. 1–12. DOI: 10.1145/3173574.3174106.

[58] Y. Zhang and X. Chen, 'Explainable recommendation: A survey and new perspectives,' *Foundations and Trends in Information Retrieval*, vol. 14, no. 1, 2020. DOI: 10.1561/1500000056.

[59] L. Cui, Y. Ge, X. He and Y. Zhang, 'Prompt-based generative news recommendation,' in *Proceedings of the 31st ACM International Conference on Information Knowledge Management (CIKM)*, ACM, 2022. DOI: 10.1145/3511808.3557220.

[60] X. Liu *et al.*, 'Continuous prompt vectors for interpretable recommendations,' *Proceedings of the 2023 IEEE International Conference on Data Mining (ICDM)*, pp. 1234–1243, 2023.

[61] X. Liu *et al.*, 'Leveraging chatgpt for coherent and contextualized explanations in recommender systems,' *Proceedings of the 2023 ACM Conference on Recommender Systems (RecSys)*, pp. 1011–1021, 2023.

[62] phalium, *Goodreads recommendations needs work*, https://www.reddit.com/r/goodreads/comments/12axyst/goodreads_recommendations_needs_work/?rdt=47757, [Accessed 09-04-2025], 2023.

[63] Goodreads, *How can i improve my recommendations*, https://help.goodreads.com/s/article/How-Can-I-Improve-my-Recommendations, [Accessed 09-04-2025].

[64] Goodreads, *Why do multiple copies of the same book keep showing up in various places*, https://help.goodreads.com/s/question/0D58V00006ZPywgSAD/why-do-multiple-copies-of-the-same-book-keep-showing-up-in-various-places, [Accessed 09-04-2025].

[65] M. Wan and J. J. McAuley, 'Item recommendation on monotonic behavior chains,' in *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, S. Pera, M. D. Ekstrand, X. Amatriain and J. O'Donovan, Eds., ACM, 2018, pp. 86–94. DOI: 10.1145/3240323.3240369. [Online]. Available: https://doi.org/10.1145/3240323.3240369.

[66] M. Wan, R. Misra, N. Nakashole and J. J. McAuley, 'Fine-grained spoiler detection from large-scale review corpora,' in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 1: Long Papers*, A. Korhonen, D. R. Traum and L. Márquez, Eds., Association for Computational Linguistics, 2019, pp. 2605–2610. DOI: 10.18653/V1/P19-1248. [Online]. Available: https://doi.org/10.18653/v1/p19-1248.

[67] B. Frederickson, *Alternating least squares (als) — implicit documentation*, https://benfred.github.io/implicit/api/models/cpu/als.html, [Accessed: 2025-04-21], 2020.

[68] F. Pedregosa, G. Varoquaux, A. Gramfort *et al.*, *Scikit-learn: Machine learning in python*, 2023. [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html#k-means.

[69] W. L. Hamilton, R. Ying and J. Leskovec, 'Inductive representation learning on large graphs,' in *Advances in Neural Information Processing Systems (NeurIPS)*, [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf, 2017, pp. 1024–1034.

[70] L. Contributors, *Lingua-language-detector*, https://pypi.org/project/lingua-language-detector/, Accessed: Apr. 23, 2025.

[71] C.Noblit, *How readers pick what to read next - written word media — writtenwordmedia.com*, Accessed: 2025-03-18. [Online]. Available: https://www.writtenwordmedia.com/how-readers-pick-what-to-read-next/.

[72] N. Reimers and I. Gurevych, *Sentence transformers documentation*, https://sbert.net/, Accessed: Apr. 23, 2025.

[73] N. Reimers and I. Gurevych, *Sentence-transformers/all-minilm-l6-v2*, https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2, Accessed: Apr. 23, 2025.

[74] T. e. a. Wolf, *Distilbert/distilbert-base-uncased-finetuned-sst-2-english*, https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-sst-2-english, Accessed: Apr. 23, 2025.

[75] The Boar. 'Overrated: The limits of 5-star rating systems.' [Accessed 09-04-2025]. (2022), [Online]. Available: https://theboar.org/2022/05/overrated-the-limits-of-5-star-rating-systems/.

[76] D. Cosley, S. Lam, I. Albert, J. A. Konstan and J. Riedl, 'The impact of rating scales on user's rating behavior,' in *User Modeling 2003*, [Accessed 09-04-2025], Springer, 2003, pp. 123–132. DOI: 10.1007/978-3-642-22362-4_11. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-22362-4_11.

[77] G. Lemaître, F. Nogueira and C. K. Aridas, *Smote - imbalanced-learn documentation*, https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html, Accessed: Apr. 23, 2025.

[78] L. McInnes, J. Healy and J. Melville, *Umap: Uniform manifold approximation and projection for dimension reduction*, https://umap-learn.readthedocs.io/en/latest/, Accessed: Apr. 23, 2025.

[79] F. e. a. Pedregosa, *Pca - scikit-learn documentation*, https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html, Accessed: Apr. 23, 2025.

[80] L. van der Maaten and G. Hinton, 'Considerably improving clustering algorithms using umap dimensionality reduction technique: A comparative study,' in *Advances in Intelligent Systems and Computing*, [Accessed 09-04-2025], Springer, 2020, pp. 409–420. DOI: 10.1007/978-3-030-51935-3_34. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-51935-3_34.

[81] J. A. Chevalier and D. Mayzlin, 'Factors influencing readers' interest in new book releases: An experimental study,' *Journal of Business Research*, vol. 58, no. 12, pp. 1583–1591, 2005, [Accessed 09-04-2025]. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0304422X05000781.

[82] A. Radford, J. W. Kim, C. Hallacy *et al.*, *Learning transferable visual models from natural language supervision*, https://arxiv.org/abs/2103.00020, [Accessed 09-04-2025], 2021.