ISSN: 2473-4901
v11 n6322

A Design of Recommender Systems with Hybrid Models

Dmytro Dobrynin
dd1503@uncw.edu
Department of Computer Science

Yao Shi
shiy@uncw.edu
Cameron School of Business

Jeffrey Cummings
cummingsj@uncw.edu
Cameron School of Business

Gulustan Dogan
dogang@uncw.edu
Department of Computer Science

University of North Carolina Wilmington
Wilmington, NC 28403, USA

Abstract

The rapid growth of digital media content makes it difficult to provide timely, relevant, and personalized recommendations. In isolation, traditional recommender systems are effective, but they often struggle with data sparsity, the cold-start problem, diversity, and adaptability to changing user preferences. Systems that accurately interpret user behaviour and item characteristics are essential. Addressing these challenges requires strategies that go beyond conventional collaborative or content-based filtering alone. This study aims to address the following research question: How can hybrid approaches integrating traditional recommendation techniques with modern machine learning methods improve the personalization, diversity, and resilience of a recommender system? To this end, we developed hybrid movie recommendation models by combining collaborative filtering with content-based analysis using NLP and two-tower neural network architectures. The collaborative filtering components utilize matrix factorization to uncover latent user preferences, while natural language processing techniques extract semantic features from movie descriptions to enhance content understanding. Neural Retrieval-Ranking models help to further refine recommendations by learning compact representations of users and items, enabling efficient and adaptive candidate selection. The evaluation methodology included both offline algorithmic performance measurement and user-centered assessments. The findings demonstrate the efficacy of selected hybrid strategies for personalized recommendations across similar application domains.

Keywords: Recommender Systems, Hybrid Filtering, Content-based Filtering, Collaborative Filtering, Natural Language Processing.

ISSN: 2473-4901
v11 n6322

A Design of Recommender Systems with Hybrid Models.

Dmytro Dobrynin, Yao Shi, Jeffrey Cummings and Gulustan Dogan

1. INTRODUCTION

With the widespread adoption of the Internet in homes and on mobile devices, the issue of information and media content oversaturation has become increasingly prominent. The sheer volume of available choices now far exceeds users' needs, making it challenging to filter and prioritize content for efficient and timely delivery. Recommender systems address those challenges by leveraging user-specific data to identify and deliver the most relevant content—particularly in streaming services and online retail platforms—ensuring users receive what they truly need.

Many commercial enterprises, such as Amazon, TripAdvisor, and IMDb, have successfully integrated recommender systems into their platforms. Unlike Netflix, which primarily focuses on suggesting films and TV series, these companies offer a broader range of products and services, supported by more diverse catalogues. This highlights the adaptability and versatility of recommender systems, which are not limited to a single domain. They can effectively guide users toward discovering books, exploring travel destinations, or even adopting innovative new technologies, all tailored to individual preferences.

However, designing and evaluating recommender systems remain persistent challenges. Several researchers (Cremonesi et al., 2010; Konstan & Riedl, 2012) argue that users are less concerned with precise rating predictions and more interested in whether the system can effectively recommend items that align with their needs and preferences. Moreover, some deep learning models, such as NeuMF and DeepFM, while powerful, are often overly complex and require large datasets to perform optimally. These models may not outperform lightweight alternatives in resource-constrained environments, where low integration and deployment time are critical. In addition, large language models (LLMs), as emerging AI techniques, demonstrate advanced capabilities in understanding linguistic complexities. Yet, they cannot fully replace the existing recommender systems, as users' preferences are derived not only from textual data but also from behavioral patterns (Li et al., 2023; Zhao et al., 2024).

This disconnect highlights the need for design and evaluation metrics that better reflect real-world user satisfaction and engagement within resource-constrained environments.

Building on the foundational concepts, definitions, and applications of recommender systems, this research aims to explore and evaluate methodologies for designing, developing, and assessing recommender systems using hybrid filtering techniques. The study places particular emphasis on implementing and comparing the accuracy and effectiveness of several custom models, guided by the principle of combining well-discovered and analysed approaches to achieve peak performance (Burke, 2002). Hybrid models, which synthesize multiple recommendation strategies, usually show an ability to mitigate individual model limitations and make the resulting system more robust and, in the case of recommenders, provide more accurate, relevant, effective, and manipulation-resistant systems to satisfy user needs (Konstan & Riedl, 2012). This research addresses common challenges in recommender systems, including the cold-start problem, new-user limitations, and the need for frequent model retraining, to enhance overall system reliability and user satisfaction.

This research evaluates the performance of the mentioned models and concludes which model turned out to be the most accurate and able to satisfy more users than others. Additionally, it identifies potential improvements, as well as the enhancements that the proposed approach could offer to existing and offered hybrid models in the future.

The study is organised into the following sections. First, the literature review section introduces the popular methods of building recommender systems, their disadvantages and advantages, and how hybrid approaches can be applied to address emerging problems. The following sections, prototype design, implementation, and evaluation, demonstrate the conception of each prototype, the implementation process for the prototypes, and the procedure for testing and assessing the prototypes. Finally, the evaluation findings, potential improvements and overall outcome are presented in the discussion, future

work, and conclusion sections.

2. LITERATURE REVIEW

There are three main traditional approaches to building recommender systems: collaborative filtering, content-based filtering, and hybrid filtering. Each approach is distinguished by its underlying logic, data used and methodology for generating recommendations.

Collaborative Filtering

Collaborative filtering (CF) is a widely used recommendation technique that suggests items to users by leveraging the preferences and behaviours of other users with similar tastes (Isinkaye et al., 2015; Papadakis et al., 2022).

CF is typically implemented using two primary approaches: user-based and item-based methods. User-based CF identifies similar users based on their historical preferences and recommends items that those users have liked (Isinkaye et al., 2015). In contrast, item-based CF calculates similarities between items based on user ratings and recommends items that are similar to those the target user has rated highly. Both approaches rely on similarity measures, such as cosine similarity or Pearson correlation, to determine the level of alignment between users or items.

CF systems offer several advantages. One of the key strengths is the ability to generate serendipitous recommendations. This enhances user engagement and is particularly effective in domains where item content is difficult to analyze. Additionally, CF systems improve over time as user interactions accumulate, enriching the dataset and enhancing recommendation accuracy. However, CF also faces significant challenges. A prominent issue is a cold-start problem, which occurs when there is insufficient data about new users or new items, resulting in many missing values in the User-Item matrix. Data sparsity is a closely related problem, where users rate only a small number of items.

Content-Based Filtering

Content-based filtering (CBF) is a recommendation technique that provides personalised item suggestions by analysing the intrinsic attributes of items and aligning them with a user's historical preferences (Javed et al., 2021; Thannimalai & Zhang, 2021).

CBF operates by comparing new items to the items that a user has previously rated positively. This process typically involves the application of various algorithmic models to assess the similarity between items. Vector Space Models (VSM), such as Term Frequency-Inverse Document Frequency (TF-IDF) and Latent Dirichlet Allocation (LDA), are commonly used to represent textual content and project the two documents onto one of the models to calculate their similarities (Falk, 2019). These methods enable the system to predict which items are likely to appeal to a user based on their prior interactions.

CBF offers key advantages, including user independence and transparency. Unlike CF, CBF relies exclusively on the user's preferences, enabling personalized recommendations based on individual interaction history. It also addresses the cold-start problem for new items, a common limitation in CF. However, CBF has notable limitations. One major limitation is overspecialization, where the system tends to recommend items or similar ones that the user has interacted with, thereby reducing diversity and novelty. Moreover, CBF, similar to CF, struggles with data sparsity, limiting its effectiveness for users with minimal interaction history.

Hybrid Filtering Model

Given the respective strengths and limitations of the above recommender systems construction techniques, it is both practical and beneficial to develop a system that would combine them to achieve better performance with fewer drawbacks of any individual one (Burke, 2002; Thorat et al., 2015). These systems are known as hybrid systems. To address the limitations of individual strategies and leverage their strengths, hybrid systems combine content-based filtering and collaborative filtering.

Burke (2002) classifies hybrid recommender systems into seven main types: weighted, switching, mixed, feature combination, feature augmentation, cascade, and meta-level. Each type represents a distinct strategy for integrating multiple recommendation techniques. Among these, the cascade model is particularly notable for its efficiency and precision. In cascade hybridization, one recommendation method is used to generate an initial, coarse list of candidate items, which is then refined by a second method. This sequential approach avoids applying complex or computationally intensive techniques to items that are either clearly irrelevant or already well-differentiated. As a result, cascade hybrids enhance both performance and accuracy by focusing on refinement efforts only where they are most

needed.

While hybridization helps alleviate cold-start limitations—particularly through the two-tower component's ability to use auxiliary features—the collaborative filtering component still requires retraining to fully incorporate new users or items.

Two-Tower Model

The two-tower model separates user and item features into two independent networks. This structure allows each network to specialize, enabling the model to learn more granular, feature-specific representations for users and items (Wang et al., 2025; Wortz & Totten, 2023). The two-tower architecture represents a significant advancement in recommender systems, addressing the limitations of traditional approaches that often rely solely on CF or CBF (Yang et al., 2020). Unlike single-model systems that focus on either user preferences or item similarity, this architecture employs two neural network "towers"—one for user features and another for item features (Varsha, 2024). This design offers several advantages. Its decoupled embedding structure enhances scalability, allowing independent and efficient training of user and item towers. It also supports diverse data integration, leveraging both structured and unstructured information to improve generalization. Additionally, the architecture enables online learning by precomputing embeddings, allowing rapid updates for new users or items without retraining the entire model. This dynamic adaptability makes the two-tower framework both efficient and responsive (Lee & Cho, 2023).

3. PROTOTYPE DESIGN

This study attempts to integrate all the above techniques into a multistage hybrid recommender system based on publicly available movie datasets. It leverages the strengths of both CF and CBF, while also incorporating the novel opportunities presented by the two-tower neural network for retrieval and ranking tasks. Our hybridisation strategies are demonstrated through the three prototypes as exhibited in Figure 1:

Prototype 1: "Simple Retrieval, Ranking, SVD and LDA"

The initial approach involves constructing a Retrieval-Ranking system based on a two-tower architecture, utilizing a limited set of features such as user and movie identifiers, along with user-movie ratings. This system is further enhanced through the integration of Singular Value Decomposition (SVD) (Klema & Laub, 1980), serving as the CF submodel, and Latent Dirichlet Allocation (LDA) (Blei et al., 2003; Chang et al., 2023; Jelodar et al., 2019), serving as the CBF submodel.

During the retrieval phase, an initial set of candidates will be selected from the entire pool of available items, and any items that the user is not interested in will be weeded out. This step can help to reduce the number of candidates — from potentially millions or tens of thousands of items to a more manageable subset. The following Ranking submodel not only reduces the number of candidate items further but also provides an estimated rating for each item, enabling them to be sorted accordingly. SVD allows to expand outputs of the Ranking submodel with items similar to those already highly rated and recommended to watch. By computing the vectors' element-wise sum (dot product) of the user embedding and the item embedding, the model generates a score for each potential item for a specific user, allowing it to generate the top k items with the highest score for the chosen user. Embedding sizes and learning rates for each model are selected accordingly to reduce training time, omit overfitting, and make the model accuracy on the test dataset better. Search space for embedding size was between 16 and 128 with step of 8. Search space for learning rate was between 1e-4 to 1e-1 with "log" sampling. For tuning, Random Search from keras_tuner was used.

After, LDA will function as a content-based model that analyses item descriptions or further features in future. Before implementing the LDA submodel, it is necessary to select a proper number of topics K. The Value K is selected by trying a range of topics from 2 to 100 and measuring coherence. After K was selected and the movie's textual descriptions were divided into topics, it will be possible to visualise those topics and the frequency of words in them.

Prototype 2: "SVD and LDA"

The second hybrid approach involves a simpler sequential flow where SVD is only followed by LDA. Initially, SVD will be applied to extract latent user preferences swiftly and find items similar to the ones with the highest ranking. Subsequently, LDA will refine these suggestions by analysing the semantic content of the movie descriptions. For example, if a user has shown an affinity for certain genres or themes through their interaction history, LDA can identify and recommend items that include similar topics.

ISSN: 2473-4901
v11 n6322

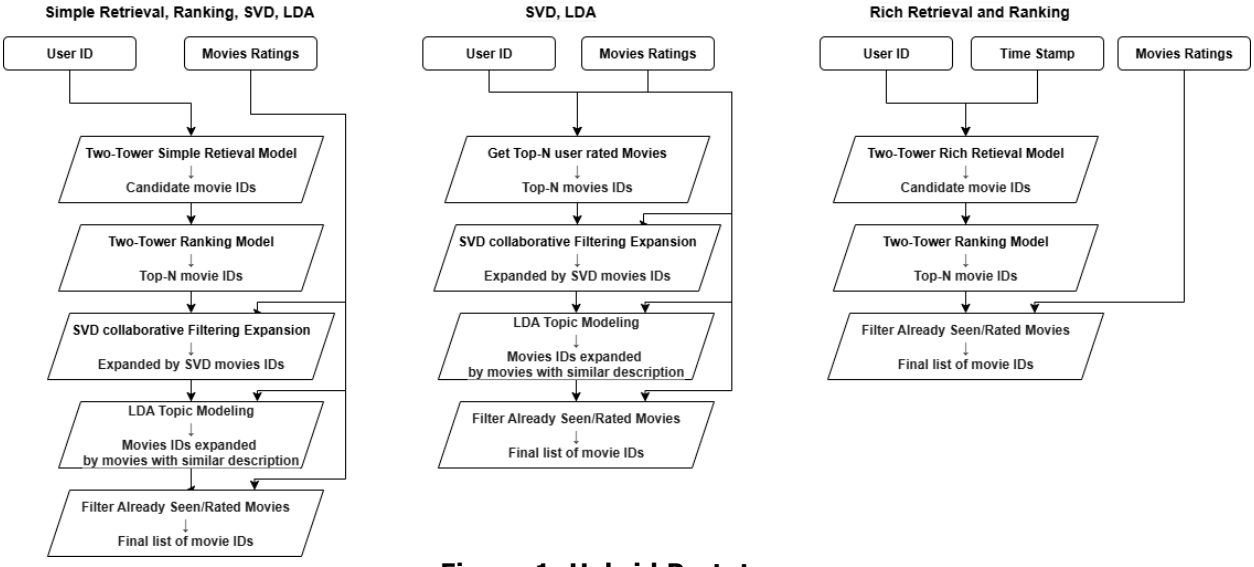

Figure 1: Hybrid Prototypes

This two-tiered method will allow us to address the shortcomings of each submodel. While SVD may overlook nuanced content features, LDA may compensate for them by adding a layer of contextual understanding, resulting in a more robust recommendation engine.

Prototype 3: "Rich Retrieval and Ranking"
The last approach enhances the first strategy by incorporating additional features into the first prototype model. These features include the timestamp and the movie's title. The architecture of the Retrieval submodel with more features. In this step, the process remains iterative after generating potential movie candidates through the Retrieval submodel with more features. Followed by ranking them using the Ranking submodel, the recommendations will be refined. This will not only allow for more personalised recommendations but also cater to the diverse range of user profiles that are prevalent in business settings.

After each stage in the studying techniques, it is essential to ensure that the films in the resultant recommendation list have not been previously viewed or reviewed by the selected user for whom the to-watch list is generated. This approach allows for evaluating how effectively the final prototypes suggest similar items without re-recommending content. When training two-tower submodels, it is crucial to prevent them from merely memorizing data from the training set; otherwise, the models may only output movies that have already been viewed or rated, leading to repetitive recommendations of items from the user's existing watch list.

4. PROTOTYPE IMPLEMENTATION

The project implementation began with collecting and processing data that was used for recommender models and their training. Most of the work was based on data found on the MovieLens website, which contains around 100 thousand ratings for 9 thousand movies (Harper & Konstan, 2015). Additional data from the same site was used to supplement the dataset for the LDA submodel. Only a table with information about 60 thousand movies from Movielen's 25M dataset was used.

Textual descriptions or plot summaries were extracted from every movie available on the IMDb website using a developed web scraper. To make the textual descriptions ready to be used for the described LDA submodel it was needed to clean texts from special symbols and punctuations and proceed to removing stop words, stemming and tokenizing the processed corpus of texts.

To find the proper K value, a series of experiments was executed to find the relationship between topic coherence and target value. The results of those experiments are depicted in Figure 2. Based on the retrieved dependency between coherence and the number of topics, it was decided to select a value K equal to 20. The reason K = 20 was chosen is that we ran experiments varying the number of LDA topics from 2 to 100, measured the topic coherence for each, and selected the value that produced the

ISSN: 2473-4901
v11 n6322

best trade-off between interpretability and coherence. According to the text, the coherence curve in Figure 2 shows that K = 20 achieved the highest (or near-highest) coherence score, making it the optimal choice for representing the movie description corpus in a way that preserves semantic clarity while avoiding over-fragmentation of topics, while also not reducing the number of topics too low. The number 20 was also reasonable to accept because it lies on the plateau of value between 15 and 22.

As was described in the methodology section, it was needed to develop and train three two-tower models for Simple Retrieval, Rich Retrieval and Ranking submodels. Each submodel requires independent training, but manually selecting optimal parameters is inefficient. To improve this process, TensorFlow and Keras tuning tools were used to identify the best hyperparameters for minimizing train and validation loss and maximizing categorical accuracy. A set of 3 parameters was selected for tuning: learning rate, embedding size and batch size. The batch size was only selected to find optimal values for simply faster training on limited computing powers. Selecting optimal hyperparameter values was essential to make sure submodels, especially two-tower ones, actually learn something. However, it was also critical not to allow them to overlearn.

Resulting values for selected parameters are shown in Table 1, and submodel training and evaluation results are shown in Table 2. The top_100_categorical_accuracy metric is a special case of the FactorizedTopK metric, which measures how often the true candidate is in the top K candidates for a given query.

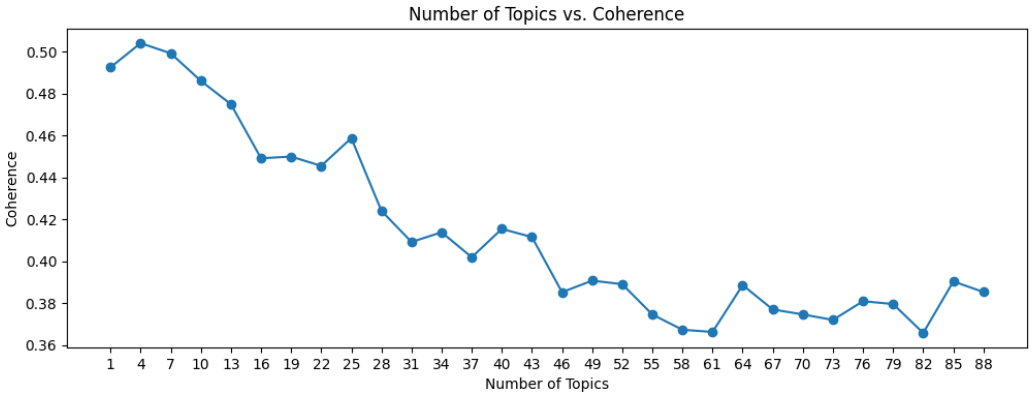

Figure 2: Topic Coherence vs. Number of Topics

Two-Tower Submodel Name	Optimal hyperparameters values
Simple Retrieval	Learning rate: 0.01 Embedding size: 64 Number of epochs before overlearning: 37
Rich Retrieval	Learning rate: 0.1 – 0.2 Embedding size:32 Number of epochs before overlearning: 4
Ranking	Learning rate: 0.1 Embedding size: 32 Number of epochs before overlearning: 20

Table 1: Optimal hyperparameter values for Two-Tower Models

Two-Tower Submodel Name	Results
Simple Retrieval	Training: top_100_categorical_accuracy: 0.2478 Validation: top_100_categorical_accuracy: 0.1163
Rich Retrieval	Training: top_100_categorical_accuracy: 0.1095 Validation: top_100_categorical_accuracy: 0.05
Ranking	Training: root_mean_squared_error: 0.8870 Validation: root_mean_squared_error: 0.8854

Table 2: Training and validation results of Two-Tower Models

As was mentioned in the Prototypes Design section, this research is focused on implementing, evaluating, and comparing 3 hybrid prototypes. The first one to implement was the "SVD and LDA". The overall process for generating recommendations using this prototype consists of a few consecutive steps. First, find the top N-rated movies by the target user, sorted by the highest average rating across all users. Second, find movies similar to them by applying an item-based approach and selecting the ones with the highest similarity score. Third, apply the LDA algorithm to the selected set of movies to expand the recommendation list with films with similar summaries or descriptions. The final step was similar to all further implemented hybrid recommenders – filter only those movies that the user had not previously watched or rated and sort them by average ratings. The output result looks like a list of recommended movie IDs` with matching movie titles.

The second prototype implemented – "Simple Retrieval, Ranking, SVD and LDA" was a synthesis of the Simple Retrieval submodel, the Ranking submodel, and the previously described SVD and LDA submodels. A Simple Retrieval submodel is initially executed, receiving a specific user ID and returning a list of movie IDs that the target user will probably find interesting. The resultant list is then passed into the Ranking submodel, which subsequently produces a list of the maximum top 10 movie IDs, sorted by the ratings determined for a specific user. The next execution process is identical to the steps described for the "SVD and LDA" model, with the only difference being input movie IDs for the SVD submodel are those generated by the Retrieval and Ranking submodels.

The last developed hybrid prototype was the "Rich Retrieval and Ranking" model. The steps are similar to those described for the Simple Retrieval submodel, except that Rich Retrieval requires two input parameters: a target user ID and a timestamp value. The timestamp value is a numerical value obtained by converting the desired date to the appropriate format. In this study, timestamp values were chosen from dates near the end of September 2018.

The output forms for each hybrid prototype are similar, with only the number of recommended movies in the final resulting list varying.

Patterns of Prototypes

After successfully implementing all three hybrid models, it may be necessary to elaborate on the specific features and negative aspects of each of them. First, the "SVD and LDA" hybrid model will generate the same results for the same user every time recommendations are asked for. This problem is easily solved by passing to the SVD submodel, not just the top-N user-rated items, which are the same every time, but a random sample of a certain size from a larger set of highly rated movies. That will make recommendations more diverse and different every time users access the system. However, to evaluate the models` results, it was decided that the recommendations list should not be changed between prototype runs. The main advantage of using LDA in reviewed hybrid models is that, unlike the SVD, it does not need to be retrained every time a new item is added to the movie directory. Because each new film comes with its own description or plot summary, similar films may be found by previously trained LDA and thus appear in any user recommendation list. Because of the mentioned 'new-user' and 'new-item' problems, the SVD cannot generate a list for a user that was never seen during model training; therefore model needs to be retrained periodically every time new data arrives. The greatest advantage was gained from adopting a two-tower architecture for the Ranking and Retrieval submodels. Those may generate at least some recommendations for any user who has never been seen by the system. Furthermore, those submodels do not require as frequent retraining as the SVD submodel, because, as mentioned in the literature review chapter, the saved embeddings are fast to compute, making updates less complex and a quicker task.

In practice, the two-tower retrieval and ranking submodels can output recommendations for new users by leveraging side information (e.g., demographic or contextual features), even in the absence of explicit rating history. However, these recommendations are generally less accurate than those for experienced users, and the SVD component must be retrained periodically to fully integrate new profiles. Thus, while the hybrid setup reduces the severity of cold-start effects, it does not completely eliminate them.

5. PROTOTYPE EVALUATION

Offline testing presents several challenges that limit our ability to perform real-time evaluation. Relying solely on offline metrics such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE)—which are ultimately derived from model outputs—may not sufficiently capture predictive accuracy. Moreover, without the capacity to conduct controlled experiments with a large user base, it becomes difficult to assess user

satisfaction or determine whether the recommended items are genuinely appreciated. As a result, accurately estimating precision and recall metrics remains a significant challenge.

Therefore, it was suggested to create or separate several imaginary user profiles with specific watch and rating histories. These profiles may include 'experienced' users as well as users with almost any prior system usage history. The approach involved simulating these users and having a group of respondents express their opinion on how each of the three presented prototypes coped with generating recommendations for a separate group of fictitious users. The target group was offered a questionnaire in which they expressed their overall satisfaction with certain models numerically and indicated the number of movies or titles from the offered recommendation list, which they believed the users would enjoy the most.

From the MovieLens dataset description, we know that any user had left at least 20 ratings from a selected set of movies. To increase the diversity of the user profiles chosen for testing and evaluation, it was decided to select five users from among 610: User 1, User 2, and User 3 are considered the most experienced users, with between 450 and 550 reviews, while Users 4 and 5 are considered the least experienced users, with only 20 ratings recorded.

A preliminary survey was carried out to assess the recommender systems. Through Google Forms, eleven respondents were asked to rate the models' overall performance (on a scale from 1 to 10, with 1 indicating disastrous performance with completely irrelevant recommendations, and 10 – top-tier performance, all recommendations are up to user taste and anticipations and provide the number (3,7,10, etc.) of well-predicted films among all the recommendations made by specific methods. Profiles that contained the viewing and rating history of each of the five test users were provided. Survey participants were required to simulate the target users based on a specified profile and decide whether they liked recommended movies from the provided generated lists or not.

Most of the respondents were graduate students, ranging in age from 23 to 27. Their knowledge of movies and the film domain varied greatly; some had little experience with watching and analysing movies, while others were highly knowledgeable in this area. As a result, those groups had the most difficulty assessing the recommender model results because they either knew too little about it or were overanalysing and complicating their conclusions, causing a possible bias.

After the review gathering process was completed, each model's precision and average rating could be computed. To determine the precision score, the average ratio of how frequently movies from generated lists were liked by a user was calculated. The survey shows the preliminary results presented in Table 3.

Prototype	Precision	Score
Simple Retrieval-Ranking, SVD and LDA	57%	6.72
SVD and LDA	53%	6.36
Rich Retrieval-Ranking	39%	4.36

Table 3: Preliminary Survey Results of Prototypes Performance

6. DISCUSSION

This study designed, analysed, and compared three hybrid movie recommender systems integrating classical content-based and collaborative filtering techniques alongside a recent two-tower neural network model. All the prototypes demonstrate the potential to generate movie recommendations for selected users.

According to the previous chapter's questionnaire results, the "Simple Retrieval-Ranking, SVD and LDA" model emerged as the most effective in providing suitable, accurate, and curated recommendations. Both average score and precision turned out to be the best. The hybrid approach and continuous refinement of intermediate results proved their adoption and yielded excellent results.

To conclude the results and determine whether there is a statistical difference between the two models, a two-sample t-test was completed. To ensure that we can apply the t-test to the data we had, it was first confirmed that the two samples of precision data have a normal distribution using the Shapiro-Wilk test, and the variances are not too different. The received p-value of 0.31 is bigger than the significance level of 0.05, meaning that we should accept the hypothesis that the mean precisions of the first two models are equal.

Interestingly, prototypes' precision was the same for experienced users, but for users who had watched the fewest movies, the difference between top prototypes was much greater, more than 10%. However, the results of the Rich Retrieval Ranking prototype were deemed

unsatisfactory. There are several possible explanations. During the review-gathering process, a few complications arose. In addition to the previously mentioned different movie backgrounds of respondents, some people expressed their views on the fact that the movie lists generated by the "Rich Retrieval-Ranking" prototype included very obscure titles that they had no idea how to evaluate. These issues could have significantly impacted the results. A larger-scale evaluation or a change in the evaluation process may be needed to confirm the results or identify any misconceptions about them.

While the evaluation of recommendation accuracy and user satisfaction was based solely on survey responses from participants who assessed model outputs for only five test user profiles, it is essential to acknowledge the inherent limitations of this assessment approach. As a result, the obtained findings may not be entirely conclusive, and further testing and analysis may yield unexpected results, necessitating a more comprehensive evaluation of the developed models.

The second prototype, "Simple Retrieval-Ranking, SVD and LDA", yielded the most impactful results. Each submodel was sufficiently trained, allowing their combination to generate valuable movie recommendations. Moreover, the system shows promise for further enhancement with the incorporation of additional training data.

Individual testing suggested that the "Rich Retrieval-Ranking" prototype tends to overtrain, potentially due to the inclusion of the timestamp feature as one of its embeddings. Further investigation is necessary to confirm whether this factor is the root cause.

7. FUTURE WORK

The effectiveness of generating comprehensive and high-quality recommendations with the proposed prototypes is significantly constrained by the size of the initially selected dataset. Achieving higher accuracy and improved user feedback for each model would likely require a dataset several orders of magnitude larger, thereby increasing the volume of data, the number of users, and the diversity of movies.

A key avenue for improvement involves refining the two-tower model architecture. Future iterations may incorporate additional features beyond the existing timestamp and movie titles, such as user age, movie genre, and other relevant attributes. Moreover, the removal of the timestamp feature is also under consideration, though further experimentation and evaluation will be necessary to assess its impact.

Additionally, new approaches to model evaluation could be explored. One potential direction is leveraging AI-driven chatbot agents to assess the quality of the generated recommendation lists. By equipping a GPT-based agent with historical user rating data and provided recommendations, it may be possible to simulate user preferences and evaluate how well the recommended movies align with a target user's interests.

By changing the model architecture, training and test dataset sizes and evaluation approach, it might be possible to improve recommendation accuracy and enhance models' speed, versatility and flexibility.

8. CONCLUSION

This research contributes to the recommender systems domain by developing and evaluating three distinct recommendation models, two of which performed well based on precision and average rating metrics. The work highlights the effectiveness of selected hybrid modelling strategies for personalised recommendations and provides a comparative analysis that may inform future model selection in similar contexts. Additionally, the project introduces practical insights on balancing accuracy and diversity in recommendation outputs. Overall, it offers a framework and empirical findings that can support continued improvement in recommender system design.

In conclusion, the results obtained and the overall performance of "Simple Retrieval-Ranking, SVD and LDA" with "SVD and LDA" hybrid models proved their potential. However, there remains significant scope for further refinement and improvement.

9. REFERENCES

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3, 993-1022.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4), 331-370. https://doi.org/10.1023/A:1021240730564

Chang, S. Y., Wu, H. C., Yan, K., Chen, X., Huang, S. C. H., & Wu, Y. (2023). Personalized multimedia recommendation systems using higher-order tensor singular-value-decomposition. *IEEE Transactions on*

ISSN: 2473-4901
v11 n6322

Broadcasting, 70(1), 148-160. https://doi.org/10.1109/TBC.2023.3278111

Cremonesi, P., Koren, Y., & Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, 39-46. https://doi.org/10.1145/1864708.1864721

Falk, K. (2019). *Practical recommender systems*. Simon and Schuster.

Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. *ACM transactions on interactive intelligent systems (tiis)*, 5(4), 1-19. https://doi.org/10.1145/2827872

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian informatics journal*, 16(3), 261-273. https://doi.org/10.1016/j.eij.2015.06.005

Javed, U., Shaukat, K., Hameed, I. A., Iqbal, F., Alam, T. M., & Luo, S. (2021). A review of content-based and context-based recommendation systems. *International Journal of Emerging Technologies in Learning*, 16(3), 274-306. https://www.learntechlib.org/p/219036/

Jelodar, H., Wang, Y., Yuan, C., Feng, X., Jiang, X., Li, Y., & Zhao, L. (2019). Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey. *Multimedia tools and applications*, 78(11), 15169-15211. https://doi.org/10.1007/s11042-018-6894-4

Klema, V., & Laub, A. (1980). The singular value decomposition: Its computation and some applications. *IEEE Transactions on automatic control*, 25(2), 164-176. https://doi.org/10.1109/TAC.1980.1102314

Konstan, J. A., & Riedl, J. (2012). Recommender systems: from algorithms to user experience. *User modeling and user-adapted interaction*, 22(1), 101-123. https://doi.org/10.1007/s11257-011-9112-x

Lee, W. M., & Cho, Y. S. (2023). A Flexible Two-Tower Model for Item Cold-Start Recommendation. *IEEE Access*, 11, 146194-146207. https://doi.org/10.1109/ACCESS.2023.3346918

Li, L., Zhang, Y., & Chen, L. (2023). Personalized prompt learning for explainable recommendation. *ACM Transactions on Information Systems*, 41(4), 1-26. https://doi.org/10.1145/3580488

Papadakis, H., Papagrigoriou, A., Panagiotakis, C., Kosmas, E., & Fragopoulou, P. (2022). Collaborative filtering recommender systems taxonomy. *Knowledge and Information Systems*, 64(1), 35-74. https://doi.org/10.1007/s10115-021-01628-7

Thannimalai, V., & Zhang, L. (2021). A content based and collaborative filtering recommender system. In *2021 International Conference on Machine Learning and Cybernetics (ICMLC)*, IEEE, 1-7. Https://doi.org/10.1109/ICMLC54886.2021.9737238

Thorat, P. B., Goudar, R. M., & Barve, S. (2015). Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110(4), 31-36. https://doi.org/10.5120/19308-0760

Varsha, M. (2024). Two tower recommendation system. *Medium*. Retrieved August 2025 from https://medium.com/@varshamoturi3/two-tower-recommendation-system-d2da761fcbce

Wang, Y., Xiong, F., Han, Z., Song, Q., Zhan, K., & Wang, B. (2025). Unleashing the Potential of Two-Tower Models: Diffusion-Based Cross-Interaction for Large-Scale Matching. In *Proceedings of the ACM on Web Conference 2025*, 304-312. https://doi.org/10.1145/3696410.3714829

Wortz, J., & Totten, J. (2023). Scaling deep retrieval with TensorFlow Recommenders and Vertex AI Matching Engine. *Google Cloud*. Retrieved August 2025 from. https://cloud.google.com/blog/products/ai-machine-learning/scaling-deep-retrieval-tensorflow-two-towers-architecture

Yang, J., Yi, X., Zhiyuan Cheng, D., Hong, L., Li, Y., Xiaoming Wang, S., ... & Chi, E. H. (2020). Mixed negative sampling for learning two-tower neural networks in recommendations. In *Companion proceedings of the web conference 2020*, 441-447. https://doi.org/10.1145/3366424.3386195

Zhao, Z., Fan, W., Li, J., Liu, Y., Mei, X., Wang, Y., ... & Li, Q. (2024). Recommender systems in the era of large language models (llms). *IEEE Transactions on Knowledge and Data Engineering*, 36(11), 6889-6907. https://doi.org/10.1109/TKDE.2024.3392335