Enhancing Movie Recommendation Systems with Web Scraped Metadata and Hybrid Deep Learning Techniques

Griffith Baker
North Carolina State University
Raleigh, NC USA
gmbaker3@ncsu.edu

1 INTRODUCTION AND BACKGROUND

1.1 Problem

Recommendation systems play a pivotal role in providing personalized content to users in the digital media landscape. However, the increasing volume and variety of available content present challenges for these systems in terms of maintaining high prediction accuracy and user satisfaction. Our project aims to tackle the issue of enhancing a movie recommendation system by incorporating additional metadata from external sources, such as IMDB, and integrating deep learning techniques. We will explore whether these advanced methods can yield more accurate predictions and better user satisfaction across various domains than traditional techniques.

1.2 Related Work

The Netflix Prize competition in 2006 sparked a surge of research in media recommendation systems as platforms, users, and content options continued to expand. The competition's dataset, containing over 100 million movie ratings from 480 thousand anonymized Netflix subscribers for more than 17 thousand movie titles, promoted this growth due to its accessibility, size, diversity, real-world relevance, and the extensive literature surrounding it [8].

The winning algorithm, developed by the collaborative team BellKor's Pragmatic Chaos, combined Matrix Factorization techniques and ensemble models to improve prediction accuracy, integrating collaborative filtering methods, temporal effects, and probe blending[7]. This groundbreaking approach not only won the competition but also set the stage for future research on recommendation systems by demonstrating the effectiveness of combining different techniques and incorporating additional data.

Subsequent research has applied deep learning techniques like Restricted Boltzmann Machines (RBMs) [1] and Autoencoders [6] to collaborative filtering problems, yielding competitive results in movie recommendation tasks. These works contributed to expanding the range of methods used in recommendation systems and showed the potential of deep learning in improving prediction accuracy.

Researchers have since explored web scraping for gathering additional data to enhance recommendation systems. In particular, incorporating external movie metadata into matrix factorization models to improve recommendation performance has been a promising avenue of research. This line of research has demonstrated the value of external data sources in boosting the performance of recommendation systems, broadening the possibilities for future improvements [5].

The Netflix Prize significantly impacted collaborative filtering research, fostering idea exchange and the development of more effective algorithms. As media platforms continue to expand, the need for efficient and accurate recommendation systems will become increasingly critical, emphasizing the importance of ongoing research and innovation in this domain.

Matrix factorization techniques proved instrumental in developing efficient algorithms for the competition. These techniques enabled quick training and prediction generation while incorporating additional data signals and views. Alongside matrix factorization, RBMs also showed promise in collaborative filtering when combined with K-Nearest Neighbors (KNN) models [4].

The competition uncovered various data effects, including binary information (considering non-random movie selection for rating), temporal effects (impacting short and long-term ratings), and the importance of blending algorithms. By the competition's end, the winning team utilized an ensemble of blends, leveraging diverse sets of predictors and blending methods to optimize prediction accuracy [7].

2 METHOD

2.1 Novel Aspects

Our Project explores the novel aspects of combining web scraped metadata with deep learning techniques in order to improve movie recommendations.

The first novel aspect of our project involves *collecting a novel dataset*. By using web scraping, we can gather additional movie information from IMDB, which will be combined with the Netflix dataset. This integration will create a richer dataset that could potentially boost the performance of our recommendation system. As shown in the Paper “The Unreasonable Effectiveness of Data” large corpora of data are extremely important for effective machine learning models [2].

This data is used to train our *Hybrid Deep Recommendation System* which is the second novel aspect of our project. What makes this network hybrid is that it incorporates both collaborative filtering and content-based methods. By using deep learning techniques and combining them with the additional scraped features, our project goes beyond the traditional recommendation systems used for the Netflix Prize Competition to potentially deliver more accurate and personalized movie recommendations.

2.2 **Approach**

Our approach involves implementing a Hybrid Deep Recommendation System that combines collaborative filtering and content-based methods. Additionally, we utilized web scraped metadata from IMBD in an attempt to enhance the predictions of the model. What follows is a general description of the techniques and final systems implemented in this project.

First, we will collect the necessary data and perform preprocessing. This involves obtaining the Netflix Prize dataset used for the 2009 Netflix Prize competition which includes user ratings, movie titles, and related information. Then we will scrape additional metadata from IMDB using web scraping techniques, with Python libraries like Requests-HTML. Both datasets will be cleaned and preprocessed to ensure the data is free from inconsistencies and that missing values are normalized. The Netflix movie ids and IMDB dataset will be merged based on the movie titles, creating a unified dataset for analysis.

For feature engineering, we will create new features based on the additional metadata gathered from IMDB. Examples include movie genres, directors, actors, runtime, descriptions, and IMDB ratings. The data will then be encoded to ensure compatibility with machine learning algorithms. For example, the description feature will be encoded using tf-idf, an algorithm that uses the frequency of words to determine how relevant those words are in a document.

The

The Hybrid Deep Recommendation System model will consist of two parts. The first is a collaborative filtering that uses deep learning techniques, such as matrix factorization with neural networks. This allows the model to recommend based on what other users have liked. The second is a content-based model using deep learning techniques, like neural networks, to learn features from the movie metadata. These two systems will be combined by concatenating both parts, followed by one or more dense layers, and finally, a regression output layer to predict user ratings.

To train and evaluate the network, we will split the datasets into training and testing sets. Additionally, a deep collaborative filtering model will be trained and evaluated using solely the Netflix dataset with no additional metadata. These two systems will be evaluated based on performance and compared against the winning algorithm developed by BellKor's Pragmatic Chaos from the Netflix Prize competition.

2.3 **Rationale**

We chose to implement a hybrid deep recommendation system as it combines the strengths of both collaborative filtering and content-based methods. In our case, collaborative filtering offers personalized recommendations based on other users’ historical preferences, while content-based methods consider movie features to provide recommendations similar to a user’s past interests based on movie metadata.

We opted to use deep learning techniques such as neural networks with hidden layers, due to their ability to capture complex patterns and nonlinear relationships in the data. Other potential approaches, such as using linear regression, ridge regression, and lasso regression were not considered due to their limited ability to capture these patterns.

We chose to concatenate the output layers of the collaborative filtering and content-based models in our hybrid system. Alternative approaches, such as blending or stacking, were not pursued as they might not fully leverage the combined strengths of both models.

We chose to use the Python Requests-HTML library to aid in web scraping as it offered a relatively simple

and efficient way to gather additional movie information from IMDB.

3 PLAN & EXPERIMENT

3.1 Datasets

The Netflix prize dataset is a well-known dataset used for the purpose of predicting user ratings for movies. The dataset was released in 2006, as part of a competition to improve the accuracy of their movie recommendation system. The dataset contains “over 100 million ratings from 480 thousand randomly chosen, anonymous Netflix customers over 17 thousand movie titles” [8]. The ratings are presented as integers, one to five. Our reasons for choosing this dataset are that it is publicly available, large and diverse, tackles real-world problems, and has a large literature backing.

The Internet Movie Database (IMDB) is a comprehensive online repository of information related to movies, television shows, actors, directors, and more. To further enhance the Netflix prize dataset, we decided to web scrape IMDB for additional metadata about the movies present in the Netflix dataset, such as movie genres, directors, actors, runtime, and descriptions. By incorporating this supplementary dataset, we aim to create a richer, more informative dataset that allows us to leverage the power of content-based methods in our Hybrid Deep Recommendation System. We hope that this additional information can potentially improve the prediction ability of our model.

3.2 Hypotheses

In this project, we will address the following main questions:

1) Can Hybrid Deep Recommendation Systems with additional metadata scraped from IMDB and Matrix Factorization with dense layers improve on the RMSE compared to the 2009 Netflix Prize-winning algorithm?

2) How does the accuracy of Deep Recommendation Systems with and without the additional metadata from IMDB compare, and does this supplementary information lower the validation error when predicting user movie ratings?

3) Can web scraping data from websites with similar information, such as IMDB, effectively enhance the prediction ability of recommendation models and demonstrate the advantage of incorporating additional training data in improving a model's performance?

By investigating these hypotheses, we aim to evaluate the potential of using web scraping to gather additional movie information from IMDB, and whether training a Hybrid Deep Recommendation System with this supplementary data can outperform a classical prediction model in terms of prediction accuracy. This research will provide valuable insights into the importance of additional training data and the potential superiority of neural networks over classical prediction models when utilizing such additional data.

3.3 Experimental Design

To answer the main questions outlined in our hypotheses, we conducted a series of experiments using our Hybrid Deep Recommendation System and compared its performance to traditional approaches, such as BellKor's Pragmatic Chaos (the winning Netflix algorithm) from 2009, which is approximately 15 years old [7].

Before conducting the experiments, we preprocessed the User-Data portion of the Netflix Prize dataset to extract all ratings and form a matrix. The file structure, a mix of JSON and CSV formats, necessitated additional cleaning and formatting. We only loaded one out of four of the User-Data files to decrease training times and ensure we could fit all the data in memory. We further filtered the user data by selecting the top one percent of users with the most ratings to drastically reduce training times and prevent Compute Unified Device Architecture (CUDA), out-of-memory errors. Finally, we shuffled the filtered dataset with a fixed random state of 42 and split it into a training set (80%) and a test set (20%) to evaluate the recommendation system's performance. When training on the machine learning models, the date column was dropped.

Next, we performed processing on the Movie Titles with metadata by dropping all movies that did not have any IMDB data. Next, we filled in missing values for the year, rating, duration, and aggregate average rating with the mean and the content rating with the mode. The genre, actors, directors, creators, keywords, and description fields were split into lists if they contained multiple entries and filled with an empty string if there was a missing value. Finally, we encoded the content rating using a label encoder, the

genre, actors, directors, and creators using a multi-label binarizer, and the keywords and description columns using a tfidf vectorizer. For the multi-label binarizer, we only kept features that had at least 100 instances and limited the tfidf vectorizer to 500 features.

For both neural networks, we used a training loop that would save the model checkpoint when the validation accuracy improved and would stop training when the validation increased five times in a row. We mounted a FUSE filesystem that was connected to the cloud to ensure that results were not lost if the machine was terminated and that we could resume training from the last checkpoint if needed. A checkpoint was taken every 5 epochs. We used mean squared error (MSE) as our loss function during training and root mean squared error (RMSE) to compare models. The Adam optimizer was utilized with a learning rate of 0.003. An embedding size of 500 and a batch size of 4096 were used for both networks.

For our first neural network, we used Matrix Factorization with hidden layers. This is the model that does not use the additional training data. The user and movie data were embedded with an embedding size of 500. Next, we took the dot product of the embedded movie and user vector and concatenated that output with the user and movie vector. After that, we used five hidden layers with our first layer having a feature size of 2 times the embedding size plus 1 and outputting 256 neurons. The following layers had outputs of 128, 64, 32, and 1 respectively. After each layer besides the output layer, the rectified linear unit (ReLU) activation function was applied and a dropout of 0.2 was used. The output was flattened and returned.

Our second neural network, the Hybrid Deep Recommendation System, used the same design as the first neural network with the addition of movie metadata. This metadata was put through a linear layer with ReLU activation whose input was the number of metadata features and output was the embedding size of 500. Next, the user vector and metadata vector were multiplied together and concatenated with the user-movie vector, user vector, and movie vector before going through the same hidden layers as the previous neural network.

For the experiments, we designed a Hybrid Deep recommendation system neural network, which combined embedding layers for users and movies with fully connected layers for processing the TF-IDF vectors. Grid search with cross-validation was used for hyperparameter tuning, optimizing the model's performance by training it on different combinations of hyperparameters and selecting the combination that yielded the best performance on the validation set. Some of the hyperparameters tuned included learning rate, batch size, and the size of the embedding.

To answer our first, second, and third question, we trained the Hybrid Deep Recommendation System to converge with the hyperparameters and preprocessing techniques discussed above and reported the root mean square error (RMSE).

To answer questions two and three, we also trained and evaluated our baseline algorithm, the winning Netflix algorithm, using the same preprocessed dataset used to train the machine learning models. The hyperparameters used for this were a sample size of 1000 and an iteration count of 700.

To be able to thoroughly answer our second question, we trained the matrix factorization model with hidden layers to convergence and reported the root mean square error (RMSE). Then we will compare the results from both deep learning models.

To compare the Hybrid Deep Recommendation System's effectiveness, we benchmarked it against BellKor’s Pragmatic Chaos algorithm, which was the Netflix Prize-winning algorithm in 2009. This allowed us to assess the deep learning-based model's performance in the context of established methods.

During the course of our experiments, we encountered several challenges. The main difficulties were managing the large volume of data and fitting it into GPU memory, ensuring reliable and consistent scraping of additional metadata, determining the most effective way to combine the collaborative filtering and content-based information, using a suitable computing machine that had an adequate GPU with a good amount of GPU memory. The GPU used for these tests was the NVIDIA GeForce RTX 2080 Ti which had 11 GBs of memory.

To overcome these difficulties, we employed the following solutions:

Data Management and GPU Memory Limitations: We scaled down the size of the User-data to only use 0.0025 of the corpus size. This allowed us to iterate on our models much faster and not have to deal with CUDA out-of-memory errors. We explored using sparse tensors for the scrapped metadata at first as

we weren't limiting the number of features produced by tfidf vectorization and the multilabel binarizer but ultimately decided to limit features instead.

Reliable Web Scraping: To ensure consistency in the scraped data, we implemented error handling in the web scraping program and also dropped all movies that did not contain scraped metadata.

Combining the IMDB metadata in the neural network: We experimented with different techniques for combining the collaborative filtering and content-based information, such as concatenation, blending, and stacking, and ultimately chose to concatenate to preserve as much data as possible.

4 RESULTS

Below are the RMSE scores for the machine learning algorithm and baseline one using the winning algorithm compared to the machine learning ones.

Bellkor Algorithm RMSE: 0.768156400414371

Machine Learning Algorithms:
Matrix Factorization with Hidden Layers RMSE: 0.6118078618023489
Deep Hybrid System RMSE: 0.5269750458848950

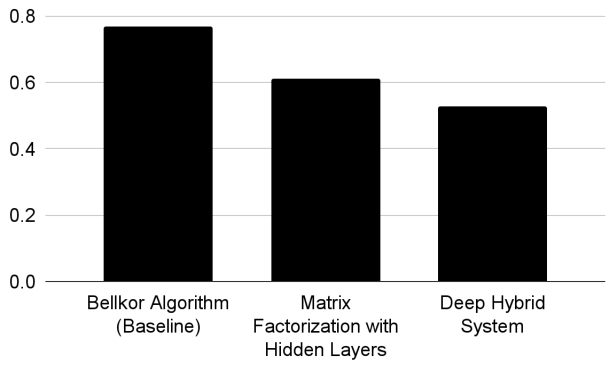

Figure 1: Root mean squared error (RMSE) for baseline and machine learning models

4.2 Discussion

The experimental results provide answers to the research questions posed.

The Hybrid Deep Recommendation Systems with additional metadata scraped from IMDB and Matrix Factorization with dense layers significantly improved the RMSE compared to the 2009 Netflix Prize-winning Bellkor Algorithm. The Bellkor Algorithm had an RMSE of 0.768156400414371, while the Matrix Factorization with Hidden Layers model had an RMSE of 0.6118078618023489, and the Deep Hybrid System had an even lower RMSE of 0.5269750458848950. These results demonstrate that incorporating additional metadata and leveraging deep learning techniques can enhance recommendation system performance.

The Deep Hybrid System, which utilized additional metadata from IMDB, outperformed the Matrix Factorization with Hidden Layers model without the supplementary information. This indicates that incorporating external metadata into the model can effectively lower the validation error when predicting user movie ratings. The difference in RMSE between the two models suggests that the additional IMDB data contributed to the improved performance of the Deep Hybrid System.

The results show that web scraping data from websites with similar information, such as IMDB, can effectively enhance the prediction ability of recommendation models. In this case, the Deep Hybrid System, which incorporated additional IMDB data, demonstrated a clear advantage over the Matrix Factorization with Hidden Layers model in terms of RMSE. This supports the notion that incorporating additional training data can improve a model's performance, confirming the findings from prior works.

Although the experimental results provide definitive answers to the research questions, there are several potential areas for further investigation. For example, it would be valuable to explore the impact of different sources of external metadata on the performance of recommendation systems. Additionally, experimenting with other deep learning techniques, such as RBMs and Autoencoders, could yield more insights into the potential improvements that can be achieved in recommendation systems. Finally, a more extensive comparison of various blending methods and ensemble techniques could help identify optimal strategies for enhancing prediction accuracy.

5 CONCLUSIONS

Throughout this project, several valuable lessons were learned, spanning various aspects of machine learning, data processing, and model optimization. Key takeaways include how to scrape metadata using

Requests-HTML, clean and merge messy data, and build neural network architecture and training pipelines. Additionally, this project provided experience in setting up a machine with the required libraries and packages for machine learning, building a neural network that incorporates metadata and understanding the workings of embedding vectors. The project also highlighted the differences between sparse and dense tensors, the importance of limiting the number of features when using tfidf vectorization and multi-label binarizer, and how to prepare data for machine learning. Lastly, the importance of a test set for optimizing hyperparameters and the superiority of the Adam optimizer over SGD were demonstrated, as well as the significance of regularization on dense layers.

Due to time constraints, several ideas were not pursued during this project. These include using sparse tensors to save memory for movie metadata information, using standard deviation or min-max scaling for movie metadata preprocessing when filling in missing values, training with a larger percentage of the dataset, and performing hyperparameter sweeps on various aspects of the models. Precomputing the movie portion of the neural network for faster inference times also remains unexplored.

An essential aspect to consider when implementing recommendation systems in real-world applications is the tradeoff between inference times for the Bellkor Algorithm and neural networks. The Bellkor Algorithm relies on a technique similar to Singular Value Decomposition (SVD), which can be computationally expensive. Neural networks, on the other hand, can be more efficient in terms of inference time, although training them may require more computational resources. This tradeoff must be weighed carefully in terms of real-world feasibility.

Additionally, scalability is a crucial factor to consider when deploying machine learning models in real-world applications, particularly in the context of large datasets like the Netflix Prize corpus. Training models on such large datasets often necessitates the use of GPUs with substantial memory, as they can significantly accelerate the training process. However, this dependency on powerful GPUs may present challenges in terms of accessibility, cost, and power consumption, particularly for smaller organizations or individuals with limited resources. Additionally, as the size of the dataset increases, the memory requirements for the model and the intermediate computations can become a bottleneck, potentially leading to longer training times and difficulties in model optimization. Therefore, future work should explore techniques to mitigate these potential scalability issues, such as distributed training, model compression, and memory-efficient representations for both input data and model parameters. Addressing these scalability concerns will be essential for ensuring the practical applicability of advanced recommendation systems in various settings and maximizing their potential impact.

Moreover, the Hybrid Model's ability to incorporate additional metadata could help address the cold-start problem often encountered in recommendation systems. By leveraging metadata to enrich the model, the Hybrid Model can generate more accurate recommendations for new users or items with limited rating history, improving the overall user experience on media platforms.

In conclusion, this study has demonstrated that Hybrid Deep Recommendation Systems with additional metadata and Matrix Factorization with dense layers can significantly improve prediction accuracy compared to the Netflix Prize-winning Bellkor Algorithm. The incorporation of external metadata, such as IMDB data, has been shown to enhance prediction ability and lower the validation error in predicting user movie ratings. Moreover, the lessons learned and ideas not pursued due to time constraints highlight potential avenues for further research and optimization in the field of media recommendation systems. Overall, this project contributes to the ongoing efforts to develop efficient and accurate recommendation systems as media platforms continue to expand, emphasizing the importance of innovation and the incorporation of additional data sources.

REFERENCES

[1] Yong-ping Du, Chang-qing Yao, Shu-hua Huo, and Jing-xuan Liu. 2017. A new item-based deep network structure using a restricted Boltzmann machine for collaborative filtering. *Front. Inf. Technol. Electron. Eng.* 18, 5 (May 2017), 658–666. DOI:https://doi.org/10.1631/FITEE.1601732

[2] Alon Halevy, Peter Norvig, and Fernando Pereira. 2009. The Unreasonable Effectiveness of Data.

IEEE Intell. Syst. 24, 2 (March 2009), 8–12. DOI:https://doi.org/10.1109/MIS.2009.36

[3] A. Hernandez-Suarez, G. Sanchez-Perez, K. Toscano-Medina, V. Martinez-Hernandez, V. Sanchez, and H. Perez-Meana. 2018. A Web Scraping Methodology for Bypassing Twitter APRestrictions. Retrieved April 11, 2023 from http://arxiv.org/abs/1803.09875

[4] Michael Jahrer, Andreas Töscher, and Robert Legenstein. 2010. Combining predictions for accurate recommender systems. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, Washington DC USA, 693–702. DOI:https://doi.org/10.1145/1835804.1835893

[5] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (August 2009), 30–37. DOI:https://doi.org/10.1109/MC.2009.263

[6] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on World Wide Web*, ACM, Florence Italy, 111–112. DOI:https://doi.org/10.1145/2740908.2742726

[7] Andreas Toscher, Michael Jahrer, and Robert M Bell. The BigChaos Solution to the Netflix Grand Prize.

[8] Netflix Prize data | Kaggle. Retrieved April 11, 2023 from https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data

APPENDIX 2. GITHUB REPO

Below is the link to our GitHub repository with information on how to set up and evaluate our models. setup_vlc.txt is the best way to get started on a GPU machine (highly recommended). Our model data was too large to upload on GitHub but there is a file in the prize_dataset folder called download_data.sh that will download it to the correct location on your machine. All model setup, training, and inference is located in the deep_learning.ipynb file inside the deep-learning folder. The given gdfuse-config file will work for any ncsu.edu google account to attach your Google Drive to the filesystem. After signing in with the link given you can copy and paste the code from the redirection URL.

GitHub Repository: https://github.com/PostsDesert/CSC-422-Netflix-Recommendations-Web-Scrapping-Ensemble-Models/tree/main