Project Report: SteamData Insights (Team 7)

Niklas Wichter[1][0009−0004−1712−4701], Nikita Ivanov[1][0009−0002−4991−8480], Ana Farinha[1,2][0009−0006−0699−4845], Fatma Koumi[1][0009−0000−2034−4300], and Nyasha Chagonda[1][0009−0008−2057−3259]

[1] Universität Mannheim, Mannheim Baden-Württemberg, Germany
[2] Universidade NOVA de Lisboa, 1099-085 Lisboa, Portugal

Keywords: Recommendation Systems · Reviews · Steam · Video-Games

1 Introduction

In the last few years, gaming platforms gained power by providing centralised access to a large portion of computer games. On Steam there are thousands of games and a new user can get lost with so many possible options. Often, users face the issue of being recommended older games or games that do not align with their preferences, leading to a less satisfying experience. These recommendations can feel outdated or irrelevant, and selecting a game based on these suggestions can result in disappointment if the game does not meet the user's expectations or interests.As such, we decided to create a recommendation system that advises new users about the games they should start playing as newcomers. The primary task is to develop a recommendation system that can effectively guide new users in selecting games from the extensive library on Steam. This system will analyze various factors, such as game popularity and user reviews to provide tailored suggestions that enhance the user experience and ensure a smooth introduction to the platform.
Research Questions: How can we effectively utilize user reviews and game metadata to generate accurate recommendations for new users?
Objective: The objective of this project is to design and implement a recommendation system that provides personalized game suggestions for new users on Steam. By leveraging user reviews and game data, the system aims to enhance the user experience, reduce the overwhelming nature of the platform's extensive game selection, and help new users quickly find games that match their interests and preferences.

2 Methodology

As this project aims to create a recommendation system that can be applied to most video game platforms, we will start this next section by explaining the steps we took to create it. To make this project reproducible, our next step is to describe the methodology used to get the data, prepare it, build the recommendation systems, and evaluate them.

2.1 Data Collection

To create our datasets we used the Steams API which let us request information about specific games and their reviews using the games ID. As such, we started by defining which games we wanted information and reviews about. As we wanted to create a recommendation system for new players, we decided to create two different datasets that would help us create a more robust model. The first one would involve 10 of the biggest and most-rated games in Steam. The other would encompass a set of smaller games that do not have as many reviews.
To define which games would go to which dataset, we decided to use a Web Crawler called Selenium to extract information about the most played games, which will be used to select the bigger games with a significant amount of reviews, and the top-seller games of specific months, that will help to select the recently released games that do not have many reviews.
As such, our first data extraction process in this project involved the most played games from the Steam website: `https://store.steampowered.com/charts/mostplayed`. Leveraging this data, we constructed a list comprised of dictionaries. Each dictionary contained the game's name and App ID. This approach allowed us to capture both established and recently popular titles for further analysis. Lastly, this list was saved as a JSON file (SteamMostPlayed.json).
We repeated the exact same procedure for retrieving top sellers. The crawled website used for this step of our extraction process is:`https://store.steampowered.com/search/?category1=998&filter=topsellers&ndl=1`. This approach enabled us to retrieve top seller of the platform steam ordered by relevance. This information was saved as a JSON file (SteamTopSellers.json).
Following, we used the Web Crawler to create a list of dictionaries (SteamMostPlayed.json) that held information about the specific game, including its rank, unique identifier, title, and peak current player count within the last 24 hours relative to the data extraction time. These new data extract included the most popular new releases of the first three months of 2024 from the following Steam websites:

1. January 2024: `https://store.steampowered.com/charts/topnewreleases/top_january_2024`
2. February 2024: `https://store.steampowered.com/charts/topnewreleases/top_february_2024`
3. March 2024: `https://store.steampowered.com/charts/topnewreleases/top_march_2024`

Lastly, we noticed that some games in the top sellers' JSON files had a unique identifier but did not have a name. As such, we used Steam's API to create a JSON file called SteamGames.json that contains all unique identifiers and games' names that are present in Steam. By comparing both, we now could join the names to the correct identifiers in all files.
Our next step was to get data about the reviews of the selected games. To do so we used Steams' API which gets the reviews of a specific game when given the

unique identifier of the game. The extracted reviews are stored in parquet files that were named using the following convention: appID_reviews_numberOfReviews.parquet. Below we can see the format of each review in the parquet files:

- **recommendationid** (string): Unique identifier for the recommendation.
- **author** (object): Contains information about the author of the recommendation.
  - **steamid** (string): Author's Steam ID.
  - **num_games_owned** (integer): Number of games the author owns on Steam.
  - **num_reviews** (integer): Number of reviews written by the author.
  - **playtime_forever** (integer): Total playtime of the author on Steam in minutes.
  - **playtime_last_two_weeks** (integer): Playtime by the author in the last two weeks in minutes.
  - **playtime_at_review** (integer): The author's playtime on the game at the time of writing the review (in minutes).
  - **last_played** (integer): Unix timestamp of the last time the author played the game.
- **language** (string): Language of the review (e.g., "english").
- **review** (string): The actual review content.
- **timestamp_created** (integer): Unix timestamp of when the review was created.
- **timestamp_updated** (integer): Unix timestamp of when the review was last updated.
- **voted_up** (boolean): Whether the author upvoted their own review.
- **votes_up** (integer): Number of upvotes the review has received.
- **votes_funny** (integer): Number of times the review was marked as funny.
- **weighted_vote_score** (string): A score representing the overall rating of the review.
- **comment_count** (integer): Number of comments on the review.
- **steam_purchase** (boolean): Whether the author purchased the game on Steam.
- **received_for_free** (boolean): Whether the author received the game for free.
- **written_during_early_access** (boolean): Whether the review was written during the game's Early Access period.
- **hidden_in_steam_china** (boolean): Whether the review is hidden on the Steam China store.
- **steam_china_location** (string): User's location for the Steam China store.

2.2 Data Exploration

Our next step was to perform data exploration. We began by loading and inspecting three datasets related to Steam games: steam games, most played, and

top sellers.SteamGames(152,351 entries, 2 features), SteamMostPlayed(99 entries, 12 features), and SteamTopSellers(5,228 entries, 12 features). By telling distinct "app-id" across the datasets, 1,187,572 entries and 22 features were identified, shedding light on the extensive diversity and characteristics of games within the Steam platform. We examined their structure, content, and identified any missing values. Next, we addressed missing values by filling them appropriately and proceeded to merge the datasets based on game names to create a unified dataset. We then created a popularity score for each game by adding the rankings from the most played dataset (rank x) and the top sellers dataset (rank y).If a game does not have a ranking in one of the datasets, the missing value is treated as 0 using fillna(0). This combined score represents the overall popularity of each game based on its performance in both datasets, with a higher score indicating greater popularity. The table 1 shows the top 5 popular games based on our calculated score. This score helped us to identify the most popular games. For exploratory data analysis, we visualised the distributions of game popularity, ratings, and prices using histograms and bar plots, and conducted correlation analysis using heat maps to understand relationships between different features.
We examined the relationship between rank in sellers and max players today. The figure 1 shows an inverse relationship between the two. Top-selling games (lower rank numbers) generally have more players today, while lower-selling games have fewer players. Most data points are concentrated among the better-selling games with a wide range of player counts, while games with higher rank numbers (worse sellers) show significantly fewer players. There are a few outliers where games with lower sales still have a high number of players.

Game ID	Name	Popularity Score
93498	Nafanya the Poltergeist	5228.0
111028	Stranded In Time	5227.0
6657	Zen Chess: Champion's Moves	5226.0
109990	VERLIES II	5225.0
53492	Magic Griddlers 2	5224.0

Table 1. Top Recommended Games

As for the exploration of Review data of games, we conducted an extensive Exploratory Data Analysis (EDA). The initial step involved consolidating multiple Parquet files into a single DataFrame, ensuring that each file's content was properly merged and attributed to the correct game using its AppId. This was followed by transforming nested dictionary columns into separate columns for easier analysis. Next, we addressed missing values and ensured data consistency by formatting columns to appropriate data types, such as converting timestamps to datetime objects and boolean fields to boolean data types. Unnecessary columns were removed to streamline the dataset. We checked for and handled duplicate rows to maintain data integrity.

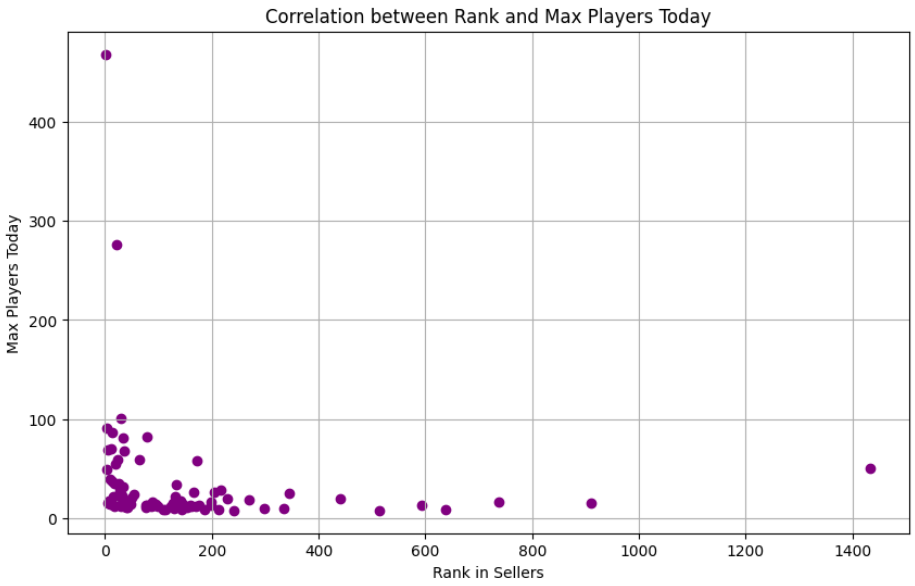

Fig. 1. Correlation between Rank and Max Players Today

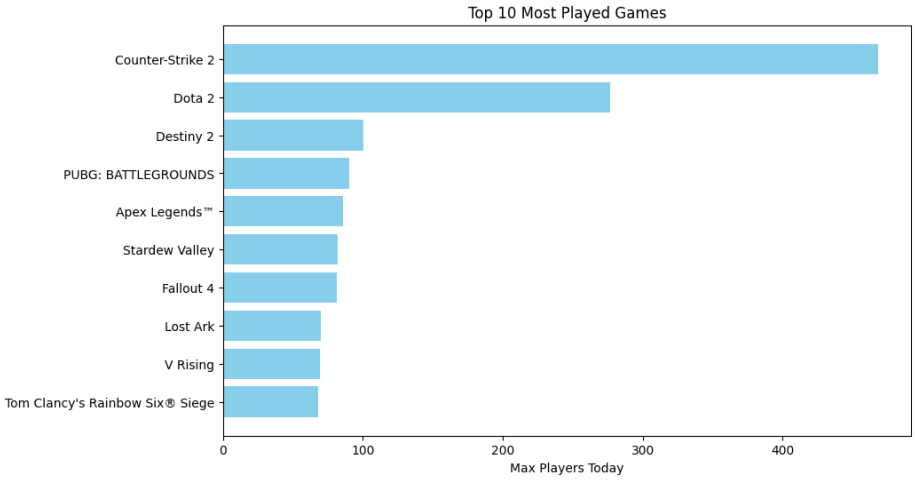

Fig. 2. Top 10 most played games

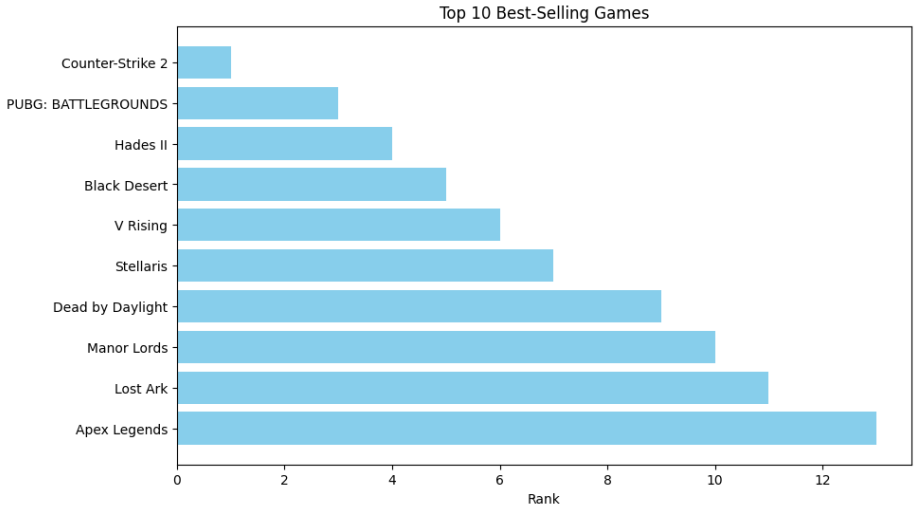

Fig. 3. Top 10 best-selling games

For numerical analysis, we isolated numerical columns and generated summary statistics to understand the central tendencies and dispersions within the data. We explored correlations between numerical features using a heatmap to identify potential relationships. Scatter matrices were created to visualize pairwise relationships between numerical variables related to user data and reviews. We examined the distribution of review timestamps to identify temporal trends and filtered data between 2022 and 2024 to analyze review activity and game contributions within this period.

We performed time series analysis to observe monthly counts of "voted up" reviews, providing insights into seasonal patterns. Additionally, we filtered and analyzed reviews starting with "EDIT:" to understand user interactions and updates. The distribution of review lengths was visualized using a histogram, and we identified the top 10 games with the longest reviews.
To further our analysis, we calculated response times between review creation and developer response, visualizing this with a histogram. Sentiment analysis was conducted using VADER to assess the polarity of reviews, and we examined the relationship between sentiment and the number of votes up using scatter plots. Text data was preprocessed to remove noise, and a word cloud was generated to visualize the most frequent terms in reviews.

2.3 Data Preprocessing

In data preparation of the games details dataset, we prepossess raw JSON files to structure the data for further analysis. We created a function 'preprocess games details', which ensures that the designated output folder exists or creates

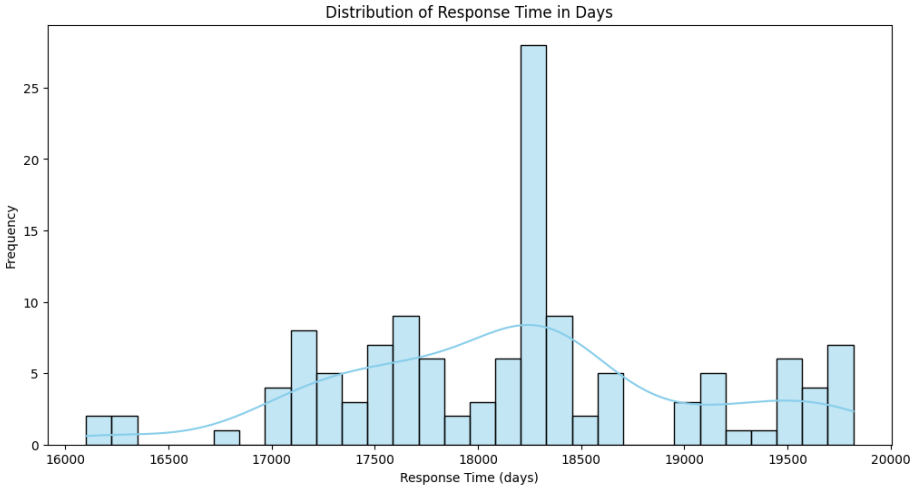

Fig. 4. Distribution of Timestamp Created (2022-2024)

a new one. This step is crucial for organizing the processed data. We then defined helper functions to assist with specific tasks: one function converts release dates from string format to timestamps, and another transforms list attributes such as genres into boolean columns. These transformations facilitate easier data manipulation and analysis. The preprocessing also involved iterating over all JSON files in the input directory. For each file, we read the JSON data and processed each row individually. We focused on extracting and standardizing key fields, including appid, is free, price, release date, number of reviews, metacritic score, usk rating, and required age'. The helper functions are then used to convert date strings and list attributes into more usable formats.Once the relevant data is extracted and transformed, we compile these preprocessed rows and save them into new JSON files in the output directory. This systematic preprocessing step is essential in creating a clean, structured dataset, ready for use in our recommendation system, ensuring that the data is consistent, accessible, and primed for analysis.

As for the preprocessing of the review data, we define a function 'preprocess review' to clean review texts by removing non-European characters and newline characters. The main preprocessing function, 'preprocess gamereviews', processes each game review by cleaning the review text, skipping entries with fewer than 20 characters, converting timestamps to datetime objects, extracting and flattening nested author information, removing unnecessary fields, and formatting boolean and float fields. We also created another function 'preprocess parquet file' which reads a parquet file, applies 'preprocess gamereviews' to each row, filters out invalid entries, adds an appid column, removes duplicates, and writes the cleaned data to a new parquet file. We then define input and output directories, clean the input directory, and iterate through each file, matching

filenames to extract app ids, and preprocess the file if it hasn't been processed already. This comprehensive preprocessing pipeline ensures that game review data is cleaned, structured, and ready for further analysis or use in our recommendation system.

2.4 Recommendation Systems

We implemented three types of collaborative filtering recommendation systems.
User Based Collaborative Filtering: We created the 'UserBasedCF' class to build a user-based collaborative filtering system. We constructed a user-item matrix where rows represent users, columns represent games and values represent ratings. We used the Nearest Neighbors algorithm with cosine similarity to find similar users. We fitted the model on the user-item matrix, and for predictions, we identified users similar to the target user and averaged their ratings for a specified game. We also implemented methods to save the trained model to a file and load it later using pickle. This system recommends games to users based on the preferences of other users with similar tastes.
Item Based Collaborative Filtering: We developed the 'ItemBasedCF' class to implement an item-based collaborative filtering system. We created an item-user matrix where rows represent games, columns represent users and values represent ratings. We also used the Nearest Neighbors algorithm with cosine similarity to find similar games. We fittled the model on the item-user matrix and for predictions we found games similar to the target game and averaged their ratings. This system recommends games to users by using the similarities between different games, suggesting items similar to those the user has rated highly.
Matrix Factorization Collaborative Filtering: We designed the MatrixFactorizationCF class to build a matrix factorization-based collaborative filtering system. We constructed a user-item matrix where rows represent users, columns represent games, and values represent ratings, filling missing values with zero. We used Non-negative Matrix Factorization (NMF) to decompose this matrix into two lower-dimensional matrices, W (user features) and H (item features). We fitted the model on the user-item matrix, and for predictions, we calculated the dot product of the corresponding user and item feature vectors from the decomposed matrices. This system captures latent features that describe both users and games, enabling us to recommend games based on underlying patterns in the data.

3 Experimental Setting

The experimental setting section lays the groundwork for evaluating the effectiveness of your recommendation system. Here, we will describe the data used to create our recommendation system along with the methods employed to evaluate it.

3.1 Datasets' Statistics

Numerical features like number of games, average review length, or distribution of release years. Mention the number of entries (rows) and features (columns) in each dataset.
In our project, we extracted and created several datasets to build an effective recommendation system for users. We gathered data on the most played games which provided insights into popular game names that consistently attacked players. This was complemented by data on top sellers highlighting games with high purchase rates. Regarding the dataset related to the reviews of the games that were created using parquet files, we came to some interesting conclusions related to some statistics of the numerical features. Below, we can see a table with the summary statistics:

	count	mean	std	min	25%	50%	75%	max
votes_up	3397339	2.5	62.93	0	0	0	1	33993
votes_funny	3397339	48040	1.4e+07	0	0	0	0	4.29e+09
weighted_vote_score	3397339	0	0	0	0	0	0	0
comment_count	3397339	0.17	0.245	0	0	0	0.49	0.99
user_num_games_owned	3397339	109.99	345.85	0	0	0	102	2516
user_num_reviews	3397339	17.33	79.57	1	2	6	15	11502
user_playtime_forever	3397339	20316	57598	0	1853	6168	16479	4201472

Table 2. Summary Statistics of Reviews Data

The summary statistics reveal significant variability and skewness in the data. Most reviews receive no upvotes or funny votes, although there are some extreme outliers. Comments are rare. On average, users own 110 games, but many have few or none. Some users write an extremely high number of reviews (up to 11,502) and have very high playtimes (up to 4,201,472 minutes). This indicates a wide range of user activities with some extreme outliers.
Additionally, we also created a correlation matrix for our numerical variables. A correlation matrix is a valuable tool for identifying potential linear relationships between variables. By analysing the correlation coefficients within the matrix, we can gain insights into how changes in one variable might influence another. In our data, we did not notice any significant correlation between our numeric variables. The biggest correlation was 0.46 between the number of votes and the quantity of up votes. Furthermore, we wanted to check the distribution of the quantity of reviews overtime. Surprisingly, we found a peak in 2024. This one is credited to the games Palworld and Helldivers 2. The last scandal which involves Helldivers 2 and Sony additionally bloated up reviews for helldivers 2 because the community was using the reviews as a lever against sony's upcoming policy.

Our next step was to concentrate more on the variable that had the reviews. We found that most of our reviews have less than 500 characters and there is an

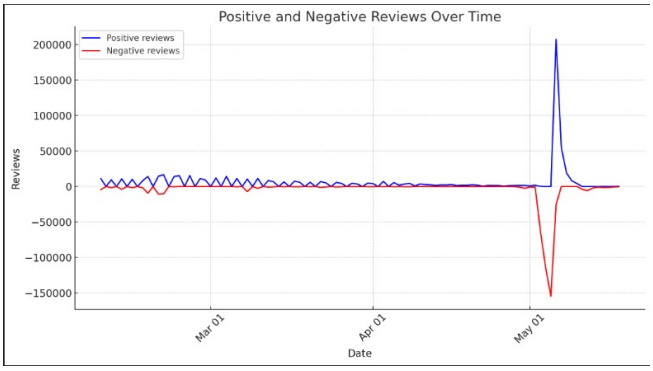

Fig. 5. Players using reviews a lever against Sony - Helldivers 2

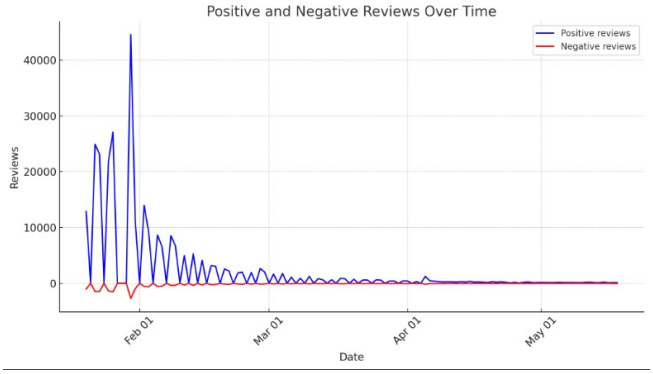

Fig. 6. Positive and Negative Reviews over time - Palworld

outlier with about 8000 characters. After, we prepossessed our reviews by lower casing every character, removing stop words, consonants and isolated consonants. With our clean reviews, we preformed sentiment analysis using Vader which is a lexicon and rule-based sentiment analysis tool. Using the compound score obtained by applying Vader, we came to the conclusion that most of our reviews were positive, some neutral and few were negative. Additionally, we created a word cloud were the words game, great, good and play standout.

3.2 Evaluation Measures

After the exploration above and creating our models, it was time to evaluate how well they performed. We evaluate the different collaborative filtering models for Steam game recommendations by leveraging user reviews and game details. We predict user ratings for games using a given model, and compare these predictions to actual ratings, and we then calculate evaluation metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Precision, Recall, and F1-score. These metrics provide a comprehensive assessment of model performance. Next, we visualize these metrics using bar charts, helping to illustrate the effectiveness of each model.

3.3 Models Performance

In this section, we summarize the evaluation results for recommendation systems on a dataset containing items with a low review count. Three recommendation techniques were evaluated: User-Based Collaborative Filtering (CF), Item-Based Collaborative Filtering, and Matrix Factorization.

The table below summarizes the evaluation results for each technique.

Technique	MAE	RMSE	Precision	Recall	F1-score
User-Based CF	7.596	124.27	0.00	0.00	0.00
Item-Based CF	8.186	122.16	0.14	0.02	0.04
Matrix Factorization	7.586	124.27	0.00	0.00	0.00

Table 3. Performance metrics of different techniques

3.4 Results Discussion

All three techniques achieved similar MAE and RMSE values, indicating comparable performance in predicting absolute rating values. Item-Based CF achieved slightly lower error metrics compared to User-Based CF and Matrix Factorization. Precision and Recall values were very low for both User-Based CF and Matrix Factorization, suggesting a high number of irrelevant recommendations.

Item-Based CF showed some improvement in precision but still low recall.

These results suggest challenges in recommending items with a low review count for all three techniques. While Item-Based CF performed slightly better in terms of error metrics, it still struggles with recommending relevant items. Further investigation is needed to explore alternative approaches or data augmentation techniques to improve recommendation accuracy for this specific scenario.

Additionally, it's important to acknowledge that the data selection process, focusing on items with a low review count ("most_played_preprocessed_low_review_count"), might be a contributing factor to the underwhelming performance across all recommendation techniques. While Item-Based CF achieved slightly better error metrics, its ability to recommend relevant items suffers due to the limited user-item interaction data available for these low-review items.

3.5 Comparison with existing solutions

During our research, we discovered an interesting project on Github[2] created by Albert Yefeng Liang and Yifan Xie, both contributors to Towards Data Science. Their project focuses on developing a comprehensive recommendation system for the Steam gaming platform. Their system is designed to enhance user experience by recommending games based on various criteria, such as popularity, quality, content similarity, and user behavior. The team implemented several algorithms to create a comprehensive recommendation system. They used a popularity-based algorithm to recommend games with the highest number of owners, providing a straightforward way to introduce new users to popular titles. The quality-based algorithm focused on games with the highest positive review rates, ensuring recommendations were based on user satisfaction rather than popularity. The content-based algorithm utilized TF-IDF and linear-kernel techniques to analyze game descriptions, recommending games with similar content to those already enjoyed by the user. Finally, they employed collaborative filtering with cosine similarity to predict user preferences based on the playing time of other users with similar tastes, and Spark ALS (Alternating Least Squares) for large-scale data to factorize the user-item interaction matrix into low-dimensional matrices for better personalized recommendations. Each method produced relevant and contextually appropriate game recommendations, enhancing the overall user experience on the Steam platform.

4 Conclusions

To improve the collaborative filtering recommendation system, some enhancements can be made. Incorporating data such as playtime and purchase history of users will capture a broader range of their preferences. Integrating contextual information, such as time of day and user location, will make recommendations

more relevant. Additionally, it would be interesting to include demographic data about users. For example, incorporating age information in the recommendation process can prevent recommending games that are too violent for younger users. This consideration of age-appropriate content can make the system more responsible and user-friendly, ensuring that recommendations align with users' suitability and preferences.

References

1. Steam Web API Overview, https://partner.steamgames.com/doc/webapi_overview, last accessed 2024/04/11
2. GitHub Repository: Steam Recommendation System, https://github.com/AlbertNightwind/Steam-recommendation-system/tree/master