Project Report for DSCI:641 - Recommender Systems

Problem Statement:

Domain: The broad domain is "Movie Recommendations," with a specific focus on enhancing user experience and engagement within the movie streaming platform.

Intended Users: The intended users of the application are movie enthusiasts or individuals seeking personalized movie recommendations. These users might be overwhelmed by the vast array of available movies and wish to discover new titles that align with their preferences and interests. Additionally, streaming platforms or movie databases could utilize such a system to enhance user engagement and satisfaction by offering tailored suggestions.

User Needs: Users want to discover new movies that match their preferences, based on their past interactions, ratings, and implicit feedback. They aim to streamline their movie selection process, minimize the time spent searching for content, and enhance their overall viewing experience by finding movies they are likely to enjoy.

Interaction with the System: Users interact with the system by providing input such as past movie ratings, genre preferences, or indicating movies they have enjoyed. The system then employs collaborative filtering or content-based recommendation algorithms to analyze user preferences and suggest relevant movies. Users may also provide feedback on recommended movies to refine future suggestions further or provide implicit feedback through their viewing history, ratings, and interactions with the platform. The system then leverages this data to generate personalized movie recommendations for the users.

Characteristics of Effective Recommendations:

Relevance: Recommendations should accurately reflect users' preferences and interests.
Novelty: Recommendations should introduce users to new and diverse content, avoiding over-recommendation of already popular or familiar movies.
Serendipity: Recommendations should occasionally introduce unexpected or serendipitous choices that pleasantly surprise users.
Timeliness: Recommendations should be provided promptly and contextually, considering the user's current needs and preferences.

User Satisfaction: Ultimately, the effectiveness of recommendations should be measured by user satisfaction, reflected in increased engagement, longer viewing sessions, and positive feedback.

Experimental Evaluation:
We use algorithms such as Matrix Factorisation, Item based collaborative filtering, Cosine Similarity, Neural Network Collaborative filtering.
we employ standard evaluation metrics like RMSE (Root Mean Squared Error) and nDCG (Normalized Discounted Cumulative Gain) to assess the accuracy and relevance of recommendations compared to users' actual preferences.

User Story:
As a hypothetical user, Sarah frequently uses the movie streaming platform after a long day at work to unwind and relax. She often finds herself spending more time scrolling through the vast library of movies than actually watching something, feeling overwhelmed by the multitude of options available. Sarah wishes she could receive personalized recommendations that align with her preferences, saving her time and effort in finding something she'll enjoy.

One evening, Sarah logs into the platform and notices a section labeled "Recommended for You" prominently displayed on the homepage. Curious, she clicks on it and finds a curated list of movies tailored to her tastes, featuring a mix of her favorite genres and actors and some intriguing new releases. Sarah selects a movie from the list and enjoys it thoroughly, appreciating how the recommendations introduced her to a film she might have overlooked otherwise. She leaves positive feedback on the platform, is satisfied with her viewing experience, and eagerly looks forward to exploring more recommendations in the future.

Dataset Description:

Ideal Data:

For the recommender application, the ideal data would include:

User Data: Information about users' demographics, preferences, viewing history, ratings, and interactions with the platform. This data would help in understanding users' tastes and preferences, allowing for personalized recommendations.

Movie Data: Detailed information about movies, including title, genre, release year, cast, crew, plot summary, and user-generated tags. This data would facilitate content-based filtering and enrich the recommendation process.

Rating Data: User ratings for movies, including the rating value and timestamp. This data would be essential for collaborative filtering approaches, enabling the system to learn from users' past interactions.

Tag Data: User-generated tags applied to movies, providing additional metadata and insights into users' categorizations and preferences.

External Data: Supplementary data from external sources such as IMDb or TMDb, including movie metadata, user reviews, and popularity rankings. Integrating this data could enhance the recommendation quality and provide context for the recommendations.

Obtaining this data would involve collecting user data through registration forms, user interactions on the platform, and permission-based access to external data sources. Data privacy and ethical considerations would be paramount, ensuring compliance with relevant regulations such as GDPR.

Features that would be valuable for recommendation include user preferences (e.g., favorite genres, actors), historical interactions (e.g., past ratings, viewing history), movie attributes (e.g., genre, cast, plot keywords), and contextual information (e.g., popularity trends, release date).

Experimental Data:

For initial experimentation, we can use the MovieLens dataset, a widely used dataset in the recommendation systems community. This dataset contains user ratings for movies along with movie metadata such as title, genre, and release year.

Why the MovieLens dataset is appropriate:

Relevance: The MovieLens dataset closely aligns with the domain of movie recommendations, providing ratings and metadata for a large collection of movies.
Size: The dataset is of a substantial size, containing millions of ratings from thousands of users on thousands of movies, enabling meaningful analysis and experimentation.
Availability: The MovieLens dataset is publicly available and widely used in research, making it accessible for experimentation.
Data Quality: The dataset is well-structured and curated, reducing the need for extensive preprocessing and cleaning.

Data Description:

- Size: The MovieLens dataset used for experimentation typically includes around 20 million ratings provided by users on a scale from 1 to 5.
- Key Features: The dataset includes user IDs, movie IDs, ratings, timestamps, movie titles, movie genres, genome scores and genome tags.
- Coverage: The dataset covers a wide range of movie genres and includes ratings for popular as well as lesser-known movies.
- Integration with Other Data: Since the MovieLens dataset is self-contained, there is no need to integrate it with other data sources. However, additional external data could be incorporated in future iterations for more comprehensive analysis and recommendations.

Files in the dataset:

Tag.csv: User-Generated Tags - Captures user-specific tagging activities on movies, offering insights into user-generated metadata.

- userId: Identifier for the user applying the tag.
- movieId: Identifier for the movie receiving the tag.
- tag: The tag applied by the user.
- timestamp: Timestamp of the tag application.

Rating.csv: User Ratings - Provides a comprehensive record of user ratings for movies, enabling the assessment of user preferences.

- userId: Identifier for the user providing the movie rating.
- movieId: Identifier for the rated movie.

- rating: User's rating for the movie.
- timestamp: Timestamp of the rating.

Movie.csv: Movie Information - Contains detailed information about movies, including title and genres, providing context for user interactions.

- movieId: Unique identifier for the movie.
- title: The title of the movie.
- genres: Genre(s) of the movie.

Link.csv: Cross-Source Identifiers - Facilitates linkage to external sources such as IMDb and TMDb, enhancing the dataset's external connectivity.

- movieId: Identifier for the movie.
- imdbId: IMDb identifier for the movie.
- tmdbId: TMDb identifier for the movie.

Genome_scores.csv: Movie-Tag Relevance - Offers a nuanced understanding of movie-tag relevance through relevance scores.

- movieId: Identifier for the movie.
- tagId: Identifier for the tag.
- relevance: A score indicating the relevance of the tag to the movie.

Genome_tags.csv: Tag Descriptions - Provides tag descriptions, enhancing the interpretability of user-generated tags.

- tagId: Identifier for the tag.
- tag: The description of the tag.

Algorithms:

Matrix Factorization using Singular Value Decomposition (SVD):

Justification: Matrix factorization methods like SVD are commonly used in recommendation systems because they can effectively capture latent factors

underlying user-item interactions. In the context of movie recommendations, SVD can help uncover hidden patterns in user ratings and identify similarities between movies based on user preferences.

> Design: We will utilize the movies dataset along with the rating dataset, merging and ensuring data consistency. After filtering for certain movies, we'll create a Surprise dataset and split it into training and test sets. The SVD model will be trained using the training data, allowing us to predict user preferences for movies by predicting ratings for user-item pairs.

Item-based Collaborative Filtering with KNN:

Justification: Collaborative filtering, particularly item-based approaches, can offer effective recommendations by leveraging similarities between items based on user interactions. Using KNN with different similarity metrics like Pearson correlation and cosine similarity can provide insights into the performance of different similarity measures in the movie recommendation context.

> Design: We will implement two variations of item-based collaborative filtering using KNN with different similarity metrics: Pearson correlation and cosine similarity. These algorithms will utilize the rating dataset to compute item-item similarities and make recommendations based on the ratings of similar items by users.

Neural Collaborative Filtering (NCF):

Justification: NCF is a state-of-the-art recommendation algorithm that leverages implicit feedback to generate personalized recommendations. It can effectively handle large-scale datasets and model complex user-item interactions, making it suitable for the movie recommendation domain.

> Design: Our approach will focus on implicit feedback, utilizing interactions between users and movies as input for the recommendation model. We will use only a subset (30%) of the ratings dataset to manage memory constraints. Train-test split will be performed based on timestamps, with the most recent reviews as the test set and prior reviews as the training set. The NCF model will be trained on the training data, enabling real-time recommendation generation based on implicit feedback.

Content-Based Filtering:

Justification: Content-based filtering addresses the cold start problem by generating recommendations based on movie features such as genres and release years, making it suitable for scenarios where user interaction data is limited or unavailable.

> Design: We will leverage movie features such as genres and release years to create item-item recommendations using content-based filtering. Cosine similarity will be used to compute similarity between movies based on their features, facilitating efficient recommendation generation. This algorithm will help address the cold start problem by providing relevant recommendations even for new or less-interacted-with items.

Overall, these algorithms are chosen based on their effectiveness in capturing user preferences, handling different types of data (explicit and implicit feedback), and addressing the cold start problem in movie recommendation systems. Testing these algorithms will provide insights into their performance and suitability for the specific requirements of the movie recommendation application.

Experiments:

The data preparation involved extracting movie features such as genres and decades to facilitate content-based filtering. To address the cold start problem, both user preferences and movie attributes were incorporated. The dataset was split into training and test sets, ensuring balance to prevent bias and enabling accurate assessment of the algorithm's performance.

To run the experiment, content-based filtering computed movie recommendations using cosine similarity measures. Specific steps included computing similarity scores between movies based on their features and generating recommendations for each user. Metrics like precision, recall, and F1-score were employed to assess recommendation accuracy, while additional analysis explored diversity, fairness, and novelty.

In terms of hyperparameter tuning, parameters like neighborhood size and learning rate were optimized to enhance algorithm performance. This process involved systematically adjusting these parameters and evaluating their impact on recommendation quality using validation data.

The findings revealed content-based filtering's high accuracy in recommending similar movies but highlighted limitations in diversity and novelty. This emphasized the importance of adopting hybrid approaches combining various recommendation techniques to enhance recommendation quality. Fine-tuning algorithms and optimizing hyperparameters emerged as crucial steps for improving performance and user satisfaction.

Overall, the experiment provided valuable insights into content-based filtering's strengths and limitations as an initial recommendation algorithm. Future efforts should focus on addressing identified limitations, adopting hybrid approaches, and continuously refining algorithms to meet user needs effectively.

Reflection:

Through this project and the class, I gained valuable insights into recommendation systems and their practical implementation. Working on the experiment allowed me to apply theoretical concepts in data preparation, algorithm design, and performance evaluation. I learned the importance of considering factors beyond accuracy, such as diversity and novelty, when assessing recommendation quality. Additionally, exploring hyperparameter tuning techniques provided valuable insights into optimizing algorithm performance.

Reflecting on the experience, I see the significance of interdisciplinary knowledge in building effective recommendation systems. The integration of machine learning, data preprocessing, and domain-specific understanding is crucial for developing robust solutions. Moreover, the iterative nature of algorithm development emphasizes the need for continuous evaluation and improvement**.**