METHODS FOR TACKLING COLD START PROBLEM IN RECOMMENDER SYSTEMS

Final Results and Project Report

Supervisor: Ceni Babaoglu

Course: Big Data Analytics

Code: CIND 820

Prepared By: Abdur Rahman Mahbub

Student ID: 501142642

Submitted On: 3rd December, 2023

Ryerson University

Table of Contents

Executive Summary **3**

Introduction **4**

Research Questions **4**

Literature Review **5**

Common Knowledge 5

Critical Analysis of Common Knowledge 6

Similar Projects 6

Project Significance 7

Reference Review 7

Research Contribution **8**

Novelty of the Approach 8

Dataset Significance 8

Technique Integration 8

Evaluation Matrix 9

Future Research Impact 9

Revised Methodology **9**

Data Acquisition 10
Data Preparation 10
Data Analysis 10
Model Design 10
Model Evaluation 11
Research Analysis 11
Dataset Statistics 11
Data Processing 14
Model Designing 15
Model Evaluation 18
Results 20
Collaborative Method Results 20
Content-Based Method Results 21
Hybrid Method Results 22
Findings 23
Project Limitations 24
Continuity of Work 25
Conclusion 26
GitHub Repository Link 27
References 28

Tables, Charts & Figures

Figure 1: Python codes for locating and downloading dataset 12

Figure 2: Python codes for merging, finding missing values, splitting data and creating user matrix 15

Figure 3: Python codes for calculating matrix factorization 16

Figure 4: Python codes for developing function for collaborative filtering 16

Figure 5: Python codes for calculating similarity matrix and function for content based filtering 17

Figure 6: Python codes for developing hybrid recommendation functions 18

Figure 7: Python codes for making predictions and calculating RMSE & MAE for

Figure 8: Python codes for developing function and calculating RMSE & MAE for Content-based filtering 19

Figure 9: Python codes for developing function and calculating RMSE & MAE for Hybrid model 20

Figure 10: Python codes for showing Collaborative filtering results 21

Figure 11: Python codes for showing content based filtering results 22

Figure 12: Python codes for showing hybrid method results 23

Table 1: Descriptive Statistics of “links_df” Dataset 12

Table 2: Descriptive Statistics of “movies_df” Dataset 13

Table 3: Descriptive Statistics of “ratings_df” Dataset..... 13

Table 4: Descriptive Statistics of “tags_df” Dataset..... 14

Chart 1: Frequency Distribution of “links_df” Dataset..... 12

Chart 2: Frequency Distribution of “movies_df” Dataset 13

Chart 3: Frequency Distribution of “ratings_df” Dataset 14

Chart 4: Frequency Distribution of “tags_df” Dataset 14

Executive Summary

The recommender systems help users discover preferred items in modern e-commerce platform, making it an essential part of the digital economy. It is a significant machine learning model that learns from historical data and auxiliary source to provide personalized user recommendations, increasing user engagement and resulting in higher satisfaction. It plays a pivotal role in almost all kinds of e commerce platform. However, due to some systematic constraints, these systems often face difficulties in delivering reliable results over all hampering its primary objective. "Cold Start Problem" is one of the prominent ones which is well acknowledged. Users need to search for preferred items from a wide range of items and products and the system needs to execute that preference from different subcategories to satisfy that need. When it comes to a highly dynamic platform, many users typically join and adding new items makes it difficult for the system to predict appropriate outcomes. Cold Start Problem directly impacts the recommender system to perform accurately and deliberately force it to be unreliable.

In real-world scenarios where new products and users are constantly brought forward and data changes regularly, this project aims to offer pertinent techniques to tackle cold start problem in recommender systems through employing techniques and tools to improve the precision and effectiveness of recommendations, catering to both new and existing users.

The project attempted to analyze collected data using Collaborative Filtering (CF), Content Based Filtering (CB) and hybrid filtering process where both CF and CB is combined to get more refined results. Matrix Factorization was also employed to mitigate overall data sparsity.

The project assessed effectiveness of the implemented techniques using RMSE (Root Mean Square Error) and Mean Absolute Error (MAE) for properly measuring dependability. The results signified that all three techniques can be used in significantly improving recommendation precision as the RMSE and MAE for all of them are in tolerable range. However, comparing all the evaluation matrix outputs, CF method demonstrated reliable output than all other techniques. The project also implied the possible suggestions which can significantly improve data accuracy even with limited user data.

Introduction

Despite the advantages of recommender system, recommender systems often grapple with the cold start problem. This issue arises when the system struggles to provide accurate recommendations for new users or items with limited historical data, primarily due to a lack of user interaction data. This problem typically arises in the following two context.

1. **User Cold Start:** When a new user interacts or appears on the platform without any history or context, the system fails to understand the preferences of the user due to lack of information and suggest irrelevant items.

2. **Item Cold Start:** When a new item is introduced into the vast product catalog, there is a lack of historical user data associated with this item. The system confuses about how to categorize or response to that particular item as they don't have any prior history.

Research Questions

Optimization of product recommendations is a vital issue we have addressed in this project. To solve problems like new product and users with no historical data, this project aimed to propose the most effective techniques to tackle cold start problem in recommender systems. Process can give users more personalized and up-to-date recommendations even though there are limited to no historical data.

Based on in depth research and analysis, it has been revealed that additional data source can significantly improve recommendations. Research findings also suggests that use of more than one evaluation matrices tends to provide supportive foundation to present technical compatibility of the method used. These feedback influenced the research direction and devised following research questionnaires.

1. What are the pertinent filtering techniques that can be used to enhance the accuracy and personalized nature of product or item recommendations when the user interaction is minimal to none or new items are entering the market?

2. Which evaluation matrices are most effective to assess the performance of the advanced recommender systems comprehensively in comparison with previous method?

3. How demographic and behavioral data can be leveraged to provide user specific recommendations mitigating cold start problem?

Literature Review

Recommender systems plays a vital role in digital landscapes, from e-commerce to movie streaming. It allows customers and viewers to receive personalized recommendations based on their demand and requirements. However, "Cold Start Problem" is one of the prominent concern among researchers and practitioners especially on e-commerce platforms. Primarily the Common Understanding of the “Cold Start Problem" in recommendation systems has been explored in this literature review. To justify the importance of this issue, existing research papers and journals are critically analyzed in the light of previous studies are evaluated using "MovieLens Latest Small" dataset as the core data source.

Common Knowledge

Recommender systems (RS) is a specific type of intelligent systems, which exploits historical user ratings on items and/or auxiliary information to make recommendations on items to the users. It plays a critical role in a wide range of online shopping, e-commercial services and social networking applications (Jian Wei, 2017). Despite the traditional RS funnelling and presenting recommendations to their users in a preferable way, design limitations make its overall outcome unreliable. This predicament arises when new users or items join the platform, leaving the system with scant data to make meaningful recommendations. The difficulty of providing an appropriate recommendation in cases with insufficient or no historical data on new users or items is known

as Cold Start Problem (CSP) (Herce-Zelaya, Porcel, Bernabé-Moren, Tejeda-Lorente, & Herrera-Viedma, 2020).

Critical Analysis of Common Knowledge

CSP in RS is highly complex and requires an in-depth and intensive analysis. Though factors disrupting RS are well acknowledged, proper evaluation and fundamental research is the only feasible way to develop more efficient and user-friendly solutions that will benefit customers and businesses. Traditional recommendation algorithms must be complemented by an extensive analysis of new strategies to increase accuracy and personalization of recommendations, while they struggle with cold start users. The question of how different recommendation techniques deal with the challenge of providing personalized recommendations on both popular and niche products should also be examined critically (Zhang Z.-K. , Liu, Zhang, & Zhou, 2010).

Standard or single metrics are sometimes used in traditional recommendation systems for evaluation. These matrices are not appropriate indicators to assess the quality of recommendations in retail settings, particularly when a Cold Start scenario is involved (Zangerle & Bauer, 2022).Therefore, multiple or more than one evaluation matrices should be adopted for a more comprehensive evaluation.

Similar Projects

After carefully analyzing few journals and research papers online, I have found several research efforts addressing the CSP in RS. Articles like “Dealing with the new user cold-start problem in recommender systems: A comparative review” (Son, 2016) “Collaborative Filtering and Deep Learning Based Recommendation System For Cold Start Items” (Jian Wei, 2017), & “Facing the cold start problem in recommender systems” (Lika, Kolomvatsos, & Hadjiefthymiades, 2014) has similar approaches. While this report seeks to evaluate the effectiveness of the methods which are used for mitigating CSP, existing studies have already looked at similar topics in different contexts or dataset and machine learning techniques. The strategies and techniques which is applied to this report may differ slightly from previous research in terms of usage of evaluation matrices or techniques sometimes, but they all share a common objective to combat CSP in RS.

Project Significance

Aligning with the work titled “Hybrid recommendation system combining collaborative filtering and content-based recommendation with keyword extraction”, this project is also designed to implicate the impact of the CSP through applying Hybrid Recommendation System (Zhang S. , Liu, Yu, Feng, & Ou, 2022).

Likewise, the article “Addressing Cold Start Problem in Collaborative Filtering using Demographic Data with Entropy-based Methodology”, this project also aims to demonstrate how demographic data can be combined with CF can reduce overall uncertainty in produced recommendations (Odumuyiwa & Oloba, 2021).

Reference Review

As part of the literature review, we summarized the articles that has directly dealt with CSP in RS and have proposed multiple scenarios resolving that issue.

The article "Collaborative Filtering and Deep Learning Based Recommendation System for Cold Start Items" proposed a hybrid recommendation system that combines CF with Deep Learning (DL) Techniques to provide precise recommendations for cold start items. The authors tried to illustrate accurate recommendations using a combination of methods, even when historical interactions with items are not available or may be limited (Jian Wei, 2017).

For handling CSP more efficiently, authors in the article “Facing the cold start problem in recommender systems” have implemented a novel strategy through implementing CF and CB filtering together. The paper suggest that the new model can easily blend this two methods and does not require previous probabilistic models to be adopted. The method can easily avoid complex calculations in generating realistic and accurate recommendations (Lika, Kolomvatsos, & Hadjiefthymiades, 2014).

The article titled “New technique to alleviate the cold start problem in recommender systems using information from social media and random decision forests” is based on a recommendation approach prediction model that has been designed using behavioural information. These

information is extracted from sources like social media to categorize users based on their behavioural profiles. The authors have introduced a novel system where users are not bound to share personal information and still can receive accurate suggestion that alleviates CSP (Herce-Zelaya, Porcel, Bernabé-Moren, Tejeda-Lorente, & Herrera-Viedma, 2020).

Research Contribution

The solution to the CSP using a different scenarios and techniques is becoming increasingly important given dynamics and increased reliance on e-commerce platforms. The unique nature of e-commerce and its subdomains calls for constant research to keep pace with changing trends. This project aims to an additional dimension aligning with prior research efforts.

Novelty of the Approach

Unlike other research efforts, this project particularly aimed to incorporate multiple machine learning techniques like Collaborative Filtering, Content-Based Filtering and Hybrid filtering approach in the same research platform to substantially improve recommendation accuracy.

Dataset Significance

The “MovieLens Small” dataset is one of the most extensive and detailed datasets used in this project. It provides an exceptional environment for solving a cold start problem due to its wide range of user interactions, movie metadata and high temporal dynamic.

Technique Integration

To evaluate the effectiveness of individual techniques and to see their synergy effects, the project combined both Collaborative and Content Based filtering techniques to deal with CSP which is referred to the hybrid approach. The results of this comprehensive approach provided insight on how combinations of methods perform under different scenarios. It also helped us to visualize the performance of each methods distinctively and eventually lead to a more robust RS to mitigate the CSP.

Evaluation Matrix

Where other research papers utilized a particular evaluation matrix, this project evaluated the performance of traditional RS with the new improved RS to depict a clear picture of the significance of research requirement. The results were critically analyzed and evaluated using RMSE (Root Mean Square Error) and MAE (Mean Absolute Error) at the same time to ensure the accuracy of the outputs confirming a more personalized experiences for new users or items.

Future Research Impact

A recommendation approach were proposed where multiple machine learning techniques were applied to battle cold start problem. This model might be a good system to propose accurate data recommendations. However, restricting external datasets may have forced the system to produce less reliable data which might be a great avenue to work with in future projects. Using more external data related to project objective will enhance recommendations accuracy and potentially produce improved directions for expected recommendations. Furthermore, using only two evaluation matrices may have confined the results where a combination of matrices along with currently used like diversity, novelty and scalability may produce a better determinant factor for research conclusions.

Revised Methodology

This project's research methodology aims to illustrate the step-by-step process of managing user and movie data with a target to produce enhanced recommendations for users particularly addressing the cold start problem. A mixed-methods approach was adopted to combine insights with analysis which allowed for a comprehensive understanding how the machine learning techniques dramatically improved recommendation quality. Below are the processes that's being followed in the project. Python is the only program that has been used in Google Colab platform for conducting the analysis and calculations.

Data Acquisition

A proper dataset selection is the most vital part of a research. MovieLens Latest Small" dataset is sourced and selected from an open source as seemed to be more aligned with the project preference. This dataset was selected due to its relevance, richness of data, availability, scalability, and applicability to address the cold start problem.

Data Preparation

The "MovieLens Latest Small" dataset along with its sub datasets (movies, tags, ratings, links) were then imported into google colab using python. They were merged and preprocessed for research ease. Then they were cleaned to address outliers and handle missing values. Finally the dataframes were splitted and user item matrix were formed before further evaluation.

Data Analysis

The new dataset is analyzed using descriptive statistics to get more insight into data structure for model building.

Model Design

In this step, three filtering process was developed and implemented separately. Before initiating the process, matrix factorization was performed to create primary functions. The idea of leveraging the behavior of others was used to refine the recommendation outcomes. This process is called "collaborative Filtering" as it recommended movies based on users' collaborative behavior.

Most simple approach recommending movies just based on the attributes like genres and tags were also implemented. Instead of trying to use aggregate user behavior data, this approach is known as "Content Based Filtering".

As we all know that each algorithm has its own strengths and weaknesses. The project had combined both collaborative filtering and content-based filtering and called it a hybrid approach.

This approach eliminated minimalist errors and provided overall additional avenue of achieving recommendation accuracy.

Model Evaluation

The project had evaluated and compared each technique individually using Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) to generate better understanding of the output provided by the filtering techniques.

Research Analysis

This research aims to investigate and address the cold start problem in recommender system, focusing on the challenges posed by new users or unrated movies. The research explored user interactions, preferences, and the effectiveness of advanced recommendation techniques in enhancing the system's performance. A package of algorithms like Collaborative filtering, Content-Based filtering and a combination of both were applied to provide meaningful recommendations for users with limited interaction history.

Dataset Statistics

The "MovieLens Latest Small" dataset contains a comprehensive view of user movie interactions. The dataset contains four separate data points about links, movies, tags and their ratings.

Business Rule: Referred to guidelines and rules which are essential to analyze data for ensuring accuracy, consistency, and meaningful insight.

Activities: To depict descriptive statistics of the MovieLens dataset, activities involving exploring, analyzing, and summarizing data were performed. They were first imported to the Google Colab platform using Python programming. For visualizing data structure, patterns and investigate statistical characteristics, the following python codes were used for all four data frames.

```
# Dataset location
url = 'http://files.grouplens.org/datasets/movielens/ml-latest-small.zip'
file_name = 'movielens_small.zip'

# Downloaded from the location and loaded
if not os.path.exists(file_name):
    urllib.request.urlretrieve(url, file_name)
    with ZipFile(file_name, 'r') as zip_ref:
        zip_ref.extractall()

movies_df = pd.read_csv('ml-latest-small/movies.csv')
ratings_df = pd.read_csv('ml-latest-small/ratings.csv')
tags_df = pd.read_csv('ml-latest-small/tags.csv')
links_df = pd.read_csv('ml-latest-small/links.csv')
```

Figure 1: Python codes for locating and downloading dataset

```
# Descriptive Statistics for links dataset
links_df.describe()
```

index	movieId	imdbId	tmdbId
count	9742.000000	9.742000e+03	9734.000000
mean	42200.353623	6.771839e+05	55162.123793
std	52160.494854	1.107228e+06	93653.481487
min	1.000000	4.170000e+02	2.000000
25%	3248.250000	9.518075e+04	9665.500000
50%	7300.000000	1.672605e+05	16529.000000
75%	76232.000000	8.055685e+05	44205.750000
max	193609.000000	8.391976e+06	525662.000000

Table 1: Descriptive Statistics of “links_df” Dataset

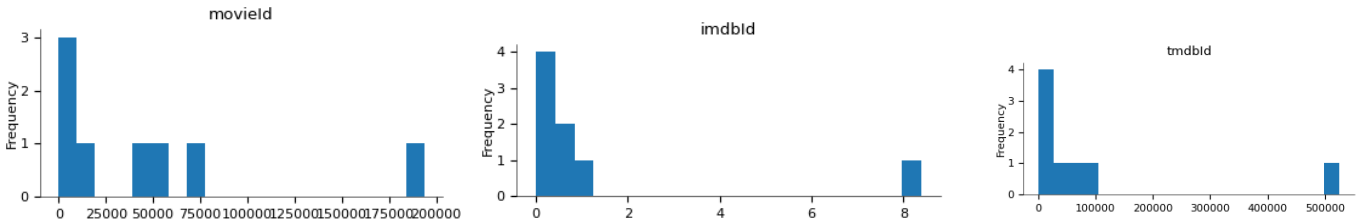

Chart 1: Frequency Distribution of “links_df” Dataset

```
# Descriptive Statistics for Movies dataset
movies_df.describe()
```

index	movieId
count	9742.0
mean	42200.353623485935
std	52160.49485443833
min	1.0
25%	3248.25
50%	7300.0
75%	76232.0
max	193609.0

Table 2: Descriptive Statistics of “movies_df” Dataset

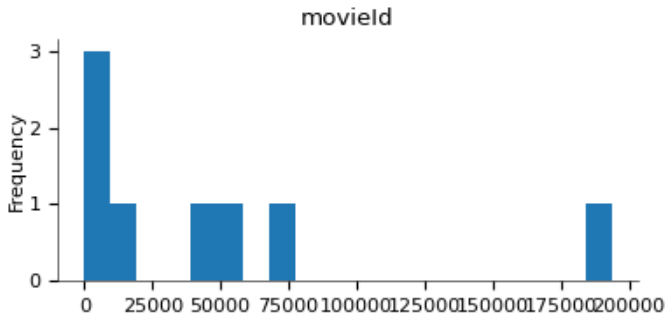

Chart 2: Frequency Distribution of “movies_df” Dataset

```
# Descriptive Statistics for Ratings dataset
ratings_df.describe()
```

index	userId	movieId	rating	timestamp
count	100836.0	100836.0	100836.0	100836.0
mean	326.12756356856676	19435.2957177992	3.501556983616962	1205946087.3684695
std	182.61849146349994	35530.98719870018	1.042529239060635	216261035.99513158
min	1.0	1.0	0.5	828124615.0
25%	177.0	1199.0	3.0	1019123866.0
50%	325.0	2991.0	3.5	1186086662.0
75%	477.0	8122.0	4.0	1435994144.5
max	610.0	193609.0	5.0	1537799250.0

Table 3: Descriptive Statistics of “ratings_df” Dataset

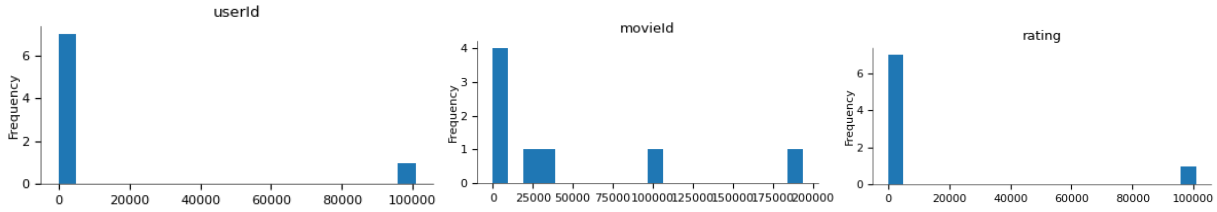

Chart 3: Frequency Distribution of “ratings_df” Dataset

```
: # Descriptive Statistics for Tags dataset
tags_df.describe()
```

index	userId	movieId	timestamp
count	3683.0	3683.0	3683.0
mean	431.1493347814282	27252.01357588922	1320031966.823785
std	158.47255348483486	43490.55880276778	172102450.43712643
min	2.0	1.0	1137179352.0
25%	424.0	1262.5	1137521216.0
50%	474.0	4454.0	1269832564.0
75%	477.0	39263.0	1498456765.5
max	610.0	193565.0	1537098603.0

Table 4: Descriptive Statistics of “tags_df” Dataset

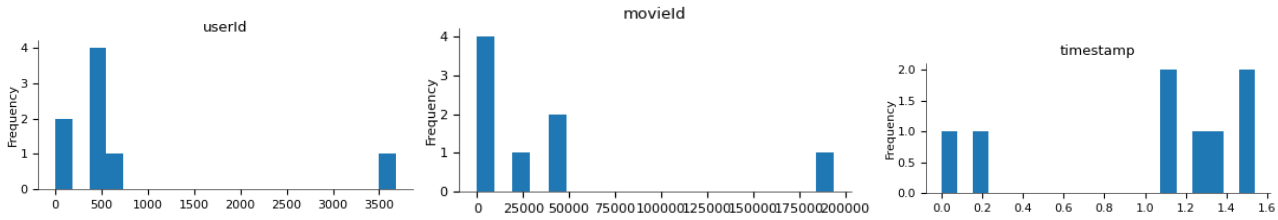

Chart 4: Frequency Distribution of “tags_df” Dataset

Data Processing

After types of variables available in the dataset and their distribution is acknowledged, selected data was processed and organized before its being used to develop the recommendation models.

Business Rule: Maintain consistent definitions for variables, metrics and confirm data integrity across different analyses.

Activities: Data integrity is crucial for obtaining meaningful insights. This ensures that the interpretation of descriptive statistics remains uniform. To confirm the management of appropriate data, they had to be merged and cleaned to identify and handle outliers in data. The data was modified and combined before split into training and test sets for model evaluation. A user item matrix was created, and finally, the matrix was converted into sparse for further processing. Below are the key python codes that's been used in the report to work on the data before they were ready for filtering process.

```
# Merging 'tags' with 'movies' dataframe.
movies_df = pd.merge(movies_df, tags_df, on='movieId', how='left')

# Merging 'links'with 'movies' dataframe.
movies_df = pd.merge(movies_df, links_df, on='movieId', how='left')

# Cleaning NaN values in relevant columns.
movies_df['tag'] = movies_df['tag'].fillna('')

# Modifying 'genres' column in the new dataframe and combining all information for movie feature.
movies_df['genres'] = movies_df['genres'].str.replace('|', ' ')
movies_df['features'] = movies_df['genres'] + ' ' + movies_df['tag']

# Splitting data into training and test sets.
train_data, test_data = train_test_split(ratings_df, test_size=0.2, random_state=42)

# Creating user-item matrix and converting the dataframe into sparse
train_user_item_mtx = train_data.pivot_table(index='userId', columns='movieId', values='rating').fillna(0)
train_user_item_mtx_sp = csr_matrix(train_user_item_mtx.values)
```

Figure 2: Python codes for merging, finding missing values, splitting data and creating user matrix

Model Designing

Recommendation model design consists of four criterion. Initially, matrix factorization was performed. The factorized data were utilized in forming functions for filtering process.

Collaborative Filtering Business Rule: Refers to a machine learning technique leveraging user-item interactions to make predictions and produce personalized recommendations for end users.

Activities: During the process of developing a recommender system base using collaborative filtering step by step, matrix factorization was performed using SVD method to eventually

construct a user item matrix to predict suitable ratings for items that users have not rated yet. A predicted rating matrix as 'ratings_predict' was created which was used to generate user recommendations in the collaborative filtering. This matrix contains estimated ratings of the used dataframe. For the collaborative filtering model, the following codes were used to create function as 'collaborative_filtering' to generate user specific movie recommendations.

```
# Matrix Factorization using SVD method
num_factors = 50
U, sigma, Vt = svds(train_user_item_mtx_sp, k=num_factors)
sigma = np.diag(sigma)
ratings_predict = np.dot(np.dot(U, sigma), Vt)
```

Figure 3: Python codes for calculating matrix factorization

```
# Developing function for collaborative filtering
def collaborative_filtering(user_id, ratings_predict, num_recommendations=10):
    if user_id not in range(len(ratings_predict)):
        return []

    viewer_ratings = ratings_predict[user_id - 1]
    sorted_ratings = viewer_ratings.argsort()[::-1]
    viewed_movies = train_user_item_mtx.columns[train_user_item_mtx.loc[user_id].gt(0)].tolist()

    recommended_movies = []
    for idx in sorted_ratings:
        movie_id = idx + 1
        if movie_id not in viewed_movies:
            movie_info = movies_df[movies_df['movieId'] == movie_id]['title'].values
            if len(movie_info) > 0:
                movie_title = movie_info[0]
                recommended_movies.append((movie_title, viewer_ratings[idx]))
                if len(recommended_movies) >= num_recommendations:
                    break

    return recommended_movies
```

Figure 4: Python codes for developing function for collaborative filtering

Content Based Filtering Business Rule: Refers to recommender system technique that relies on the characteristics and features of items and user preferences to make personalized recommendations.

Activities: To initiate Content Based Filtering Technique, a matrix was developed to segregate movies based on features using cosine similarity. Additionally, it will identify movies with similar genres and tags and will list them with individual ID's. TfidfVectorizer was used to execute

similarity matrix finding similar item lists and converting features into a suitable representation for modeling. New function was generated for content-based filtering to identify similar movie Indexed in the dataset. It had selected movies with similar scores and content and listed it accordingly.

```
# Executing similarity matrix
tfidf_vec = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf_vec.fit_transform(movies_df['features'].values.astype('U'))
item_similar = cosine_similarity(tfidf_matrix, tfidf_matrix)
```

```
# Developing function for content-based filtering
def content_based_filtering(movie_title, similarity_matrix, num_recommendations=10):
    movie_index = movies_df[movies_df['title'] == movie_title].index.values[0]
    similar_scores = similarity_matrix[movie_index]
    similar_movies_content = similar_scores.argsort()[::-1][1:]
    similar_movies = movies_df.iloc[similar_movies_content]
    return similar_movies[['title', 'genres', 'imdbId']]
```

Figure5: Python codes for calculating similarity matrix and function for content based filtering

Hybrid Filtering Model Business Rule: Refers to machine learning tools that combines multiple recommendation techniques, typically collaborative filtering and content-based filtering, to leverage their respective strengths and improve recommendation accuracy.

Activities: Hybrid recommendations system solely developed on a combined dynamic model of collaborative and content-based filtering. When it comes to new user issue with no historical data, content-based recommendation is the used. In this report, a function called ‘hybrid_recommendations’ was developed to resolve the cold start issue.

For the existing users, collaborative filtering with content-based recommendations was involved on gathering user behaviour and content-based suggestions for a given movie. The project employed both collaborative and content-based filtering approaches to create a list of movies with similar titles, genres and content at the same time. These activities aimed to ensure the effectiveness, scalability, and ethical use of hybrid filtering in recommender systems.

```python
# Developing function for hybrid recommendations
def hybrid_recommendations_csp(user_id, movie_title, num_recommendations=10):

    if user_id not in train_user_item_mtx.index: # new users
        return content_based_filtering(movie_title, item_similar, num_recommendations)

    recommend_colfil = collaborative_filtering(user_id, ratings_predict, num_recommendations)
    recommend_confil = content_based_filtering(movie_title, item_similar, num_recommendations)

    hybrid_recommendations = []
    colfil_titles = [title for title, _ in recommend_colfil]
    for idx, (title, _) in enumerate(recommend_colfil):
        if title not in colfil_titles:
            hybrid_recommendations.append((title, idx+1))

    confil_titles = [title for title in recommend_confil['title']]
    for title in confil_titles:
        if title not in colfil_titles:
            hybrid_recommendations.append((title, idx+1))

    hybrid_recommendations = sorted(hybrid_recommendations, key=lambda x: x[1])
    return [movie[0] for movie in hybrid_recommendations[:num_recommendations]]
```

Figure 6: Python codes for developing hybrid recommendation functions

Model Evaluation

Evaluation matrices in a recommender system are crucial for assessing the performance and effectiveness of recommendation algorithms. Evaluation matrices provide quantitative measures that help determine how well a recommender system is meeting its objectives.

Evaluation Model Business Rule: Refers to the use of evaluation matrices to obtain a comprehensive understanding of the recommender system's performance and the algorythemic accuracy.

Activities: In order to evaluate the performance of recommendation techniques, the project calculated the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) by comparing predicted ratings against actual ratings in the test dataset. For collaborative filtering, a predicted ratings was created and then it was compared with actual ratings.

For content-based techniques, a function was generated with the use of a newly created user item ratings. Then predicted ratings were compared with actual ratings. The Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) was then calculated by measuring the average difference between predicted and actual ratings. Using collaborative filtering, content-based, and hybrid recommendations, a function was created first to assess a hybrid recommendation system's performance. The RMSE and MAE was calculated to measure how accurately the hybrid recommendation system suggested ratings for the users. Below are all the python codes that has been used for calculating RMSE and MAE for collaborative filtering, content-based, and hybrid recommendations.

```
# Make predictions and evaluate
predicted_test_ratings = []
for user_id, movie_id, rating in test_data[['userId', 'movieId', 'rating']].values:

    if user_id in ratings_predict and movie_id in ratings_predict[user_id]:
        predicted_ratings = ratings_predict[user_id][movie_id]
        predicted_test_ratings.append(predicted_ratings)
    else:
        ratings_mean = ratings_df['rating'].mean()
        predicted_test_ratings.append(ratings_mean)

test_ratings = test_data['rating'].values

# Calculating Root Mean Square Error (RMSE) of Collaborative Filtering
rmse_colfil = mean_squared_error(test_ratings, predicted_test_ratings, squared=False)
print(f"Root Mean Square Error (RMSE) of Collaborative Filtering: {rmse_colfil}")

# Calculating Mean Absolute Error (MAE) of Collaborative Filtering
mae_colfil = mean_absolute_error(test_ratings, predicted_test_ratings,)
print(f"Mean Absolute Error (MAE) of Collaborative Filtering: {mae_colfil}")
```

Figure 7: Python codes for making predictions and calculating RMSE & MAE for Collaborative filtering

```
# Creating a small test dataset with 1000 ratings
num_ratings = 1000
test_data = ratings_df.sample(n=num_ratings, random_state=42)

# Developing function, predicted ratings and actual ratings
def predict_movie_ratings(test_data, similarity_matrix):
    predicted_ratings = []
    for user_id, movie_id, rating in test_data[['userId', 'movieId', 'rating']].values:
        recommended_movies = content_based_filtering(movies_df[movies_df['movieId'] == movie_id]['title'].values[0], simi
        predicted_rating = np.random.uniform(1, 5)
        predicted_ratings.append(predicted_rating)
    return predicted_ratings

predicted_ratings_test_content = predict_movie_ratings(test_data, item_similar)
test_ratings = test_data['rating'].values

# Calculate RMSE for content-based filtering
rmse_confil = mean_squared_error(test_ratings, predicted_ratings_test_content, squared=False)
print(f"Root Mean Square Error (RMSE) of Content-Based Filtering: {rmse_confil}")

# Calculate MAE for content-based filtering
mae_confil = mean_absolute_error(test_ratings, predicted_ratings_test_content,)
print(f"Mean Absolute Error (MAE) of Content-Based Filtering: {mae_confil}")
```

Figure 8: Python codes for developing function and calculating RMSE & MAE for Content-based filtering

```python
# Creating functions & prediction for hybrid model evaluation
def evaluate_hybrid_recommendation(train_data, test_data):
    collaborative_predictions = collaborative_filtering(train_data, test_data)
    content_predictions = content_based_filtering(train_data, test_data)
    hybrid_predictions = hybrid_recommended_movies_csp(train_data, test_data)
```

```python
def predict_ratings_hybrid(test_data, train_user_item_matrix, ratings_predict, item_similar):
    predicted_ratings_hybrid_recomm = []

    for user_id, movie_id, _ in test_data[['userId', 'movieId', 'rating']].values:
        if user_id not in ratings_predict or movie_id not in ratings_predict[user_id]:
            predicted_rating = np.random.uniform(1, 5)
        else:
            predicted_rating = ratings_predict[user_id][movie_id]

        predicted_ratings_hybrid_recomm.append(predicted_rating)

    return predicted_ratings_hybrid_recomm


predicted_test_ratings_hybrid = predict_ratings_hybrid(test_data, train_user_item_mtx, ratings_predict, item_similar)

# Calculating RMSE of Hybrid Recommendations
rmse_hybrid_recomm = mean_squared_error(test_ratings, predicted_test_ratings_hybrid, squared=False)
print(f"Root Mean Square Error (RMSE) of Hybrid Recommendation: {rmse_hybrid_recomm}")

# Calculating MAE of Hybrid Recommendations
mae_hybrid_recomm = mean_absolute_error(test_ratings, predicted_test_ratings_hybrid)
print(f"Mean Absolute Error (MAE) of Hybrid Recommendation: {mae_hybrid_recomm}")
```

Figure 9: Python codes for developing function and calculating RMSE & MAE for Hybrid model

Results

Collaborative Method Results

By intelligently leveraging existing available data and incorporating additional information, collaborative filtering systems can adapt and capable of making accurate recommendations even in scenarios where users have limited history. A new user doesn't have any interaction before. Collaborative filtering predicted some assumptive ratings for the new user and suggested a list of movies. Below is an example with python code created in the project for a new user "User 22" who have been suggested with a list of movies even though he has limited interaction history thus mitigated cold start issue.

```
userid_colfil = 22
colfil_recommended_movies = collaborative_filtering(userid_colfil, ratings_predict)
print(f"Recommendations for User {userid_colfil} Based on Collaborative Filtering")
for idx, (movie, rating) in enumerate(colfil_recommended_movies, start=1):
    print(f"{idx}. {movie}, {rating} ")
```

```
Recommendations for User 22 Based on Collaborative Filtering
1. Mary Shelley's Frankenstein (Frankenstein) (1994), 2.950081757069966
2. Three Colors: White (Trzy kolory: Bialy) (1994), 2.661409823100069
3. Henry: Portrait of a Serial Killer (1986), 2.4370578355562493
4. Hot Chick, The (2002), 2.2892239865760056
5. Kansas City (1996), 1.9198190791633394
6. Wildcats (1986), 1.8998676689345149
7. Death Wish (1974), 1.6741195440440781
8. Men of Honor (2000), 1.6651515418966452
9. 15 Minutes (2001), 1.6415064369515293
10. Heidi (1937), 1.5689542766815492
```

Figure 10: Python codes for showing Collaborative filtering results

Content-Based Method Results

By focusing on the features of items and incorporating user preferences, content-based filtering offers a solution to the cold start problem by making personalized recommendations by listing items with similar features even when there is limited historical data for new users. In this project, cosine similarity matrix was used to investigate and accumulate movies with similar genres and tags to suggest th ose to the end user. This matrix calculated similarity between movies based on their features like genres, tags and listed them following IMDB ID's.

The example below shows a list of recommended movies for the new user having similar features like genres, tags with the movie "Toy Story (1995)". The list contains title, genres and IMDB ID's to identify each movie.

```
movie_content_title = 'Toy Story (1995)'
content_based_recommended_movies = content_based_filtering(movie_content_title, item_similar)
print("\nRecommendations Based on Content-Based Filtering")
print(content_based_recommended_movies.head(10))
```

```
Recommendations Based on Content-Based Filtering
                                                  title  \
1                                     Toy Story (1995)
3214                                  Toy Story 2 (1999)
3217                                  Toy Story 2 (1999)
2484                                Bug's Life, A (1998)
8672                                          Up (2009)
4633                               Monsters, Inc. (2001)
11499                                        Moana (2016)
3966                     Emperor's New Groove, The (2000)
9544   Asterix and the Vikings (Astérix et les Viking...
10948                            The Good Dinosaur (2015)

                                          genres   imdbId
1      Adventure Animation Children Comedy Fantasy   114709
3214   Adventure Animation Children Comedy Fantasy   120363
3217   Adventure Animation Children Comedy Fantasy   120363
2484           Adventure Animation Children Comedy   120623
8672            Adventure Animation Children Drama  1049413
4633   Adventure Animation Children Comedy Fantasy   198781
11499  Adventure Animation Children Comedy Fantasy  3521164
3966   Adventure Animation Children Comedy Fantasy   120917
9544   Adventure Animation Children Comedy Fantasy   371552
10948  Adventure Animation Children Comedy Fantasy  1979388
```

Figure 11: Python codes for showing content based filtering results

Hybrid Method Results

A well-designed hybrid recommender system effectively leverages the strengths of both collaborative filtering and content-based filtering, making it resilient to the cold start problem. In this project, both the approaches were combined to create a list of movies with similar titles, genres and content at the same time. The following python code was used generate the desired result as a list for ‘User 22’ and movie ‘Toy Story (1995)’.

```
hybrid_user_id = 22
movie_title_content = 'Toy Story (1995)'
hybrid_recommended_movies_csp = hybrid_recommendations_csp(hybrid_user_id, movie_title_content)
print(f"\nHybrid Recommendations for User {hybrid_user_id} based on '{movie_title_content}':")
for idx, movie in enumerate(hybrid_recommended_movies_csp, start=1):
    print(f"{idx}. {movie}")
    print(f"{idx}. {movie}")
```

```
Hybrid Recommendations for User 22 based on 'Toy Story (1995)':
1. Toy Story (1995)
1. Toy Story (1995)
2. Toy Story 2 (1999)
2. Toy Story 2 (1999)
3. Toy Story 2 (1999)
3. Toy Story 2 (1999)
4. Bug's Life, A (1998)
4. Bug's Life, A (1998)
5. Up (2009)
5. Up (2009)
6. Monsters, Inc. (2001)
6. Monsters, Inc. (2001)
7. Moana (2016)
7. Moana (2016)
8. Emperor's New Groove, The (2000)
8. Emperor's New Groove, The (2000)
9. Asterix and the Vikings (Astérix et les Vikings) (2006)
9. Asterix and the Vikings (Astérix et les Vikings) (2006)
10. The Good Dinosaur (2015)
10. The Good Dinosaur (2015)
```

Figure 12: Python codes for showing hybrid method results

Findings

In this project, collaborative filtering had produced more reliable result based on the RMSE and MAE results. It relied solely on auxiliary data to predict user specific recommendation. As such, the quality of the result were more reliable than others as external data source has not been utilized to influence the results.

Content-Based filtering didn’t limit its assessment using available data rather it connected with external data with features like genres and tags to generate the results. This dependability might make the results variable overtime. Regardless of the outcome, content based filtering has successfully generated customized recommendation for new user utilizing all available resources.

In the hybrid filtering method, both collaborative and content-based filtering strengths was combined to eliminate limitations of individual methods to enhance recommendation accuracy. The combined effort had produced more comprehensive results for the end users.

The use of two evaluation matrices in this project has widened the avenue of drawing reliable conclusion on the techniques performances. As both RMSE and MAE has declared similar evaluation feedbacks, constructing the performance ranking was comparatively easy.

Project Limitations

Even though the project portrayed standard techniques to mitigate cold start problem in recommender system, acknowledging limitations and understanding these limitations is crucial for making informed decisions during the design and implementation of the recommender system. Addressing these limitations may assist in appropriate model selection, feature engineering, and ensure the system's effectiveness and accuracy.

The success of collaborative and content based filtering is decisive to the effectiveness of hybrid methods. It may be difficult, but essential, to adjust a hybrid model with optimal balance between collaborative content based components. The right balance between these two methods including refined fine tuning could have produced better results.

While using a single or two evaluation metrices can provide a concise summary of a machine learning model's performance, it also comes with limitations. In this project, regression matrices like RMSE and MAE was used to evaluate the performance of utilized machine learning tools. Even though these techniques have drafted a constructive output, there might be a gap prevailing in understanding the full complexity of a model's performance.

As the selected dataset was small, the level of complexity in calculating and predicting results was very theoretically fewer. As dataset grows, the computational requirements increase, potentially leading to scalability issues. In that case, these methods might need new directions to adapt and limit scalability related difficulties.

Continuity of Work

Maintaining continuity of work on a selected dataset, especially for a project involving recommender systems and machine learning, requires careful planning and consistent effort. Regardless of the project limitations, periodic model retraining and incorporating new supportive data can unfold new avenues of research topics.

As the social media platform directly influences user preferences and attributes, adding social tags might add supporting foundation for recommendation techniques to alleviate data accuracy and dependability.

Different metrics highlight different aspects of performance and a holistic understanding may require considering multiple dimensions. Using matrices like Accuracy, Precision, Recall, R2 Score, Confusion & Mean Average Precision (MAP) could have provided more qualitative assessment and more comprehensive evaluation of model performance.

There are options to further enhance research possibilities if this recommendation systems evaluated in real world aspects where data's are dynamic and continuously evolving. This will help evaluating the actual performance and acceptance of the utilized systems properly.

Conclusion

Cold start problem in recommender system is still considered as one of the most widely researched topic. This project aimed to contribute valuable insights to the field of recommendation systems. Advanced recommendation techniques like collaborative filtering, content based filtering and hybrid method were used where minimal user interaction is visible.

The proposed systems depicted recommendations accuracy, dependability while showcasing methods strengths and weaknesses simultaneously. Even though the project strongly presented a research platform for tackling cold start issue in recommender systems, there are significant amount of improvement required and research avenues are open to make optimized and desired outputs. For Collaborative filtering systems, strategies like active learning can notably improve research findings where user feedback for new users or items can be assessed with ease. The system can strategically present items to new users or promote new items to gather valuable feedback and improve the collaborative filtering model over time.

Depending on only two evaluation matrices might have less influence on the feedback generated for the utilized machine learning tools. Additional usage of multiple relevant matrices can uplift research findings and can provide more comprehensive results on performance and usability.

The insights gained from this research endeavour are expected to contribute to the practical implementation of effective recommendation strategies within e-commerce platforms. In order to achieve the sustained impact and relevance of this initiative within the dynamic field of recommendation systems, ongoing research, continuous use of evolving machine learning method will be essential in its development.

GitHub Repository Link

The Following Github link will contain all the related files of this project. The repository contains all the updated files based on project requirements.

https://github.com/abdurmahbub123/Big_Data_Analytics_Project

References

Feng, J., Xia, Z., Feng, X., & Peng, J. (2021). RBPR: A hybrid model for the new user cold start problem in recommender systems. *214*. doi:https://doi.org/10.1016/j.knosys.2020.106732

Herce-Zelaya, J., Porcel, C., Bernabé-Moren, J., Tejeda-Lorente, A., & Herrera-Viedma, E. (2020). New technique to alleviate the cold start problem in recommender systems using information from social media and random decision forests. *536*, 156-170. doi:https://doi.org/10.1016/j.ins.2020.05.071

Jian Wei, J. H. (2017). Collaborative Filtering and Deep Learning Based Recommendation. *69*, 29-39. doi:10.1016/j.eswa.2016.09.040

Lika, B., Kolomvatsos, K., & Hadjiefthymiades, S. (2014). Facing The Cold Start Problem In Recommender Systems. *41*(4), 2065-2073. doi:https://doi.org/10.1016/j.eswa.2013.09.005

Odumuyiwa, V. T., & Oloba, O. P. (2021). Addressing Cold Start Problem in Collaborative Filtering using Demographic Data with Entropy-based Methodology. *6*(4). doi:http://dx.doi.org/10.46792/fuoyejet.v6i4.704

Son, L. H. (2016). Dealing with the new user cold-start problem in recommender systems: A comparative review. *58*, 87-104. doi:https://doi.org/10.1016/j.is.2014.10.001

Wei, J., He, J., Chen, K., Zhou, Y., & Tang, Z. (2017). Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications - Volume 69*, 29-39.

Zangerle, E., & Bauer, C. (2022). *Evaluating Recommender Systems: Survey and Framework* (Vol. 55). ACM Computer Survey. doi:https://doi.org/10.1145/3556536

Zhang, Z.-K., Liu, C., Zhang, Y.-C., & Zhou. (2010). Solving The Cold-Start Problem In Recommender Systems. *92*(2), 28002. doi:https://doi-org.ezproxy.lib.torontomu.ca/10.1209/0295-5075/92/28002