Movie Recommendation System

I. Business Problem and Objective

MovieMingle is a movie streaming platform that has gained rapid popularity among movie lovers. So many movies are available on the platform that users have access to different types of movies from various genres, and many of them enjoy this freedom. However, the growing number of movies available also brings a big challenge to MovieMingle.

MovieMingle found that users are often confused in navigating through the various movie options available. They often struggled to decide what movie they should watch, which could even result in delays in decision-making or even leaving the platform without watching anything. A 10% drop in user retention in a span of 2 months due to this issue has negatively impacted the revenue of "MovieMingle." It is therefore imperative to address this issue within the next 3 to 5 weeks, given that revenue is a key factor in maintaining the sustainability of the MovieMingle platform, as well as being a key indicator in developing and expanding the service, as well as providing additional content and features that can enhance the user experience.

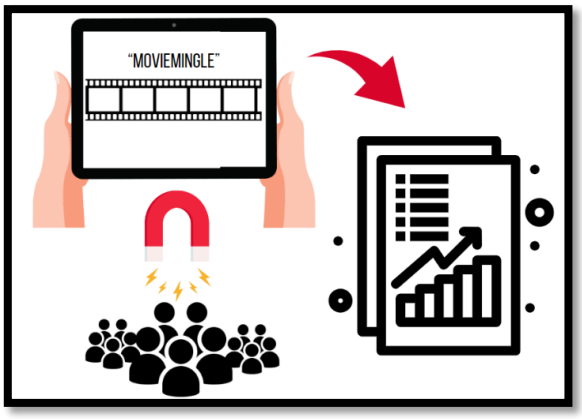

Figure 1. Illustration of MovieMingle Problem

In order to maintain and increase user activity on MovieMingle, several approaches to the problem were made in the following ways.

1. Non-ML Solution

   In the non-ML solution, a first attempt was made to create a category of "Top Popular" category that displayed the movies most watched by other users. This helped most users in finding popular movies, but many users were dissatisfied that movies they enjoyed were not included in this list. For example, a user who enjoys horror movies will be dissatisfied if the "Top Popular" list is dominated by movies from other genres such as comedy or drama. They may look for recommendations specifically in the horror genre.

   Furthermore, a second experiment was conducted by providing filters based on genre or filters based on release year through the MovieMingle feature. While this gives users more options and reduces confusion, users often still find it difficult to navigate the various options

available. In addition, over-reliance on filters based on genre or year may cause users to only interact with familiar filters, ignoring the potential to discover interesting movies outside of genre or year preferences. This is known as the "long tail" phenomenon.

Long tails are common in online markets and have enabled innovative businesses, that might not have flourished outside of long-tailed markets, to excel and become very successful.

Let us graph a market in the following way. We will order the i products by sales volume m_i, from highest to lowest, and graph the sales volume vs. the product. Note that this formulation requires that our graph be downward-sloping. Below is an example of such a graph.

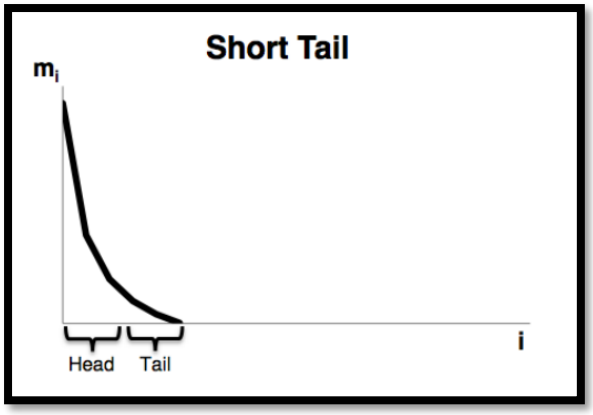

Figure 2. Short Tail
(Ashish Goel, 2009)

In the graph above, the market drops off fairly sharply to zero. This is an example of a “short tail,” or “sharp tail.” There are a small number of products with large sales volumes and a small number of products with small sales volumes. This, however, will not accurately depict many online markets. One of the defining characteristics of online markets is that they look more like the graph in Figure 3.

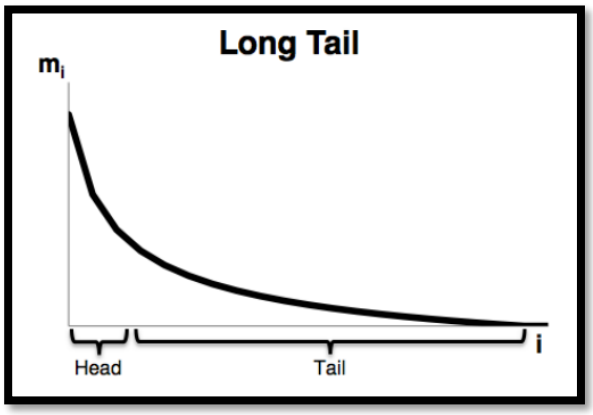

Figure 3. Long Tail
(Ashish Goel, 2009)

In Figure 3 the tail is much longer, and as such is referred to as a “long tail” or “heavy tail.” Such a distribution tells us that, in this market, there are a small number of products with high sales volumes and a very large number of products with low sales volumes. Even though the products in the tail individually have low sales volumes, the total volume in the tail is relatively large because of the large number of products it encompasses (Ashish Goel, 2009).

So the "long tail" phenomenon refers to the concept that while most product or content catalogs are centered around the most popular items or content, there are a large number of other less popular items or content (in the "long tail") that can be a significant source of sales or user consumption if given the right access and exposure. In the context of MovieMingle, there are many movies that may be lesser known but can become users' favorites if they are given the opportunity to discover them.

2. ML Solution

To address these issues more effectively, a machine learning-based solution in the form of a personalized recommendation system was implemented. This approach is divided based on the interaction of available data:

1. Implicit Data

   Implicit data is drawn from indirect user behavior such as movie clicks, time spent browsing movie pages, or adding movies to bookmarks.

2. Explicit Data

   Explicit data is drawn from direct user ratings such as movie ratings or movie reviews.

Related to this problem, there is direct interaction by users in the form of ratings. So the approach will be done using explicit data.

Furthermore, there are 2 recommendation system approaches, the first is the implementation of a recommendation system based on a content-based recommendation system. Content-based systems focus on the properties of items. The similarity of items is determined by measuring the similarity in their properties (Leskovec, 2014). This will allow MovieMingle to analyze the movies that users have seen and present more personalized recommendations based on genre, year, and other movie features that match their preferences.

Table 1. Example of Content-Based Recommendation System

Movie title	Genre	Director	Lead Actor
Inception	Sci-Fi, Action	Christopher Nolan	Leonardo DiCaprio, Ellen Page
Pulp Fiction	Crime, Drama	Quentin Tarantino	John Travolta, Samuel L. Jackson
The Dark Knight	Action, Crime, Drama	Christopher Nolan	Christian Bale, Heath Ledger

In the table above, if a user likes the movie "Inception" then the system can recommend the user with the movie "The Dark Knight" or other movies by Christopher Nolan or movies with similar genres such as action movies.

However, there are some drawbacks to this approach. Content-based recommendations are limited in suggesting movies that are very different from what the user has seen before,

which can lead to users being stuck with similar movie recommendations that they have already watched.

Therefore, the implementation of a collaborative filtering-based recommendation system is carried out. Collaborative filtering systems focus on the relationship between users and items. The similarity of items is determined by the similarity of the ratings of those items by the users who have rated both items (Leskovec, 2014).

In collaborative filtering, there are two main approaches:

- User-to-user collaborative filtering, this method focuses on the similarity between users. If there are two users who have similar preferences for items, then the system will assume that their preferences are similar for other items that have not been interacted with by either of them.

Table 2. Example of User-to-User CF Recommendation System

User	Movie A	Movie B	Movie C	Movie D
User 1	5	4	-	-
User 2	-	5	4	3
User 3	1	-	2	-
User 4	3	1	-	5

In the table above, each row represents a user who gave a rating for several movies. The movie ratings are from 1 to 5, where 5 is the highest rating. Suppose the target is user 1 and the system finds that user 2 has similar preferences to user 1. Therefore, user 2 is a similar user to user 1. So if user 2 gives a high rating to Movie C, the system will recommend Movie C to user 1 because user 1 has not watched it yet.

- Item-to-item collaborative filtering, this method focuses on the similarity between items. If two items have preferences from various users, then they are considered to be similar and can be recommended the similar item to users based on their preferences for the other item.

Table 3. Example of Item-to-Item CF Recommendation System

	User 1	User 2	User 3	User 4
Movie A	5	-	4	3
Movie B	4	5	-	4
Movie C	-	4	3	-
Movie D	-	2	-	1

In the table above, suppose the user has liked Movie A, then the system can recommend Movie B because it is similar to Movie A.

Based on the above approach, the collaborative filtering approach that will be carried out is user-to-user collaborative filtering. Due to the large amount of rating data provided by users, it is possible to find similarity patterns between users. However, when implementing User-to-User Collaborative Filtering, there will be a challenge called the "cold start problem".

The “cold-start” problem describes situations in which a recommender is unable to make meaningful recommendations due to an initial lack of ratings. This problem can significantly degrade CF performance (Schafer, 2007).

The "cold-start" problem occurs when a new user has no history of interaction with items on the platform. In this situation, Collaborative Filtering will face difficulties because there is no previous behavioral data to search for other users with similar profiles that can be used as a basis for recommendations. To solve this problem, new users who have just joined will be given recommendations based on characteristics and preferences that can be identified from the content data. From the explanation above, the user journey that will be carried out can be seen in the following figure.

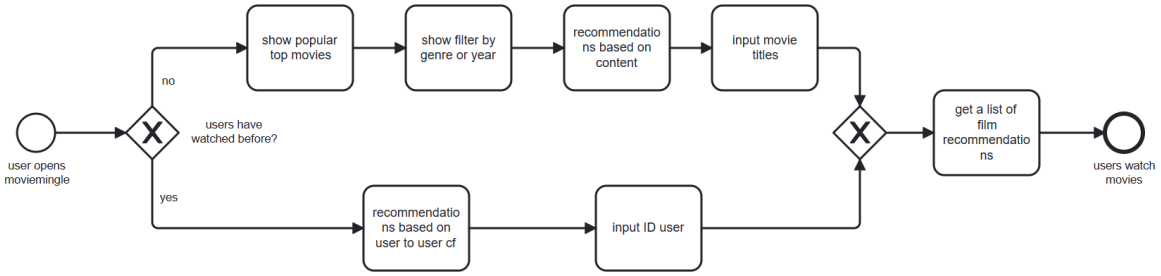

Figure 4. User Journey

II. Dataset and System Recommendation

The dataset comes from https://grouplens.org/datasets/movielens/25m/. The dataset files used are ratings.csv and movies.csv. Details of the dataset can be seen in the following table.

Table 4. Data Rating.csv

Feature	Description	Data Type
userId	User ID	int
movieId	Movie title	int
rating	Rating movies	float
timestamp	Timestamp	int

Table 5. Data Movies.csv

Feature	Description	Data Type
movieId	Movie ID	int
title	Movie title	int
genres	Genre movies	object

The data has 62,423 movies and 25,000,000 ratings. Before conducting further analysis, the data undergoes a preprocessing stage first. At this stage, we check for empty values in the rating data and movies data, the result is that there are no empty values in the data. Then the data type, feature selection, and duplicate data are checked. At the feature selection stage, the 'timestamp'

column is removed from the rating data. In addition, the results of the data type check showed that the existing data types were appropriate, and no duplicate data was found. However, since the data size is very large, the next step is to take samples from both datasets, i.e. movie data and rating data.

In developing a personalized recommendation system, a content-based recommendation approach with TFIDF is used. TFIDF is the most common weighting method used to describe documents in the Vector Space Model, particularly in IR problems. Regarding text categorization, this weighting function has been particularly related to two important machine learning methods: KNN and SVM. The TFIDF function weights each vector component (each of them relating to a word of the vocabulary) of each document on the following basis. First, it incorporates the word frequency in the document. Thus, the more a word appears in a document (e.g., its TF, term frequency is high) the more it is estimated to be significant in this document. In addition, IDF measures how infrequent a word is in the collection (Soucy, 2005).

However, content-based recommendation with the TF-IDF method has the disadvantage of being limited in identifying context. TFIDF focuses on certain keywords or attributes in the content of an item (e.g., text or metadata), but is not always able to understand the context or true meaning of the keywords. This may result in recommendations that are based on the same keywords but have different meanings, which may result in less relevant recommendations.

Next, a personalization approach in the recommendation system using user-to-user based collaborative filtering is carried out. Collaborative filtering is an approach that looks for similarities between users based on their preferences for items. In some cases, such as in linear regression models, regression can be used to model the relationship between user preferences and items, making it possible to make predictions of the ratings or scores given by users to items. The first step is to set the baseline model using the `BaselineOnly()` method from the Surprise library. `BaselineOnly` is a model that is used as a starting point or baseline in building a recommendation system.

The idea of the baseline model is to clean the data and erase effects from all the ratings such as day biases or hour biases. The Baseline prediction $\hat{r}_{ui}$ consists of many parts, global bias μ, user bias μ_u and item bias μ_i (Jahrer, 2012).

More specifically, the `BaselineOnly` algorithm models the ranking as follows:

$$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i$$

where :

- μ is the global average of ratings for all items
- b_u is a per-user effect that measures how the ratings of users, u tend to differ from the global average
- b_i is a per-item effect that measures how the rating of an item, i tends to differ from the global average.

Furthermore, three machine learning methods, namely KNN Baseline, SVD, and NMF, will be applied to find the best performance among the three. The KNN Baseline method focuses on collaborative filtering. K-nearest neighbor (KNN) is a relatively simple concept. If a point among

other points needs to be classified, KNN finds the K closest points, where K is a number defined by the user, and then assigns the class label of the majority of its neighbors. Thus, the idea is that if a point is in a neighborhood of a certain class, chances are that the point belongs to that class. The biggest advantage of the K-nearest neighbor model is its simplicity, it is relatively easy to set up and requires little maintenance. Also, the K-nearest neighbor model is conceptually very similar to how you would approach the problem manually if doing collaborative filtering because it is about grouping similar users, neighbors, together (Torstensson, 2019).

Singular value decomposition (SVD) is the most famous matrix factorization implementation. It was made famous in 2009 when it won the big Netflix competition in creating the best recommendation system, beating older K-nearest neighbor and statistical models (Koren, etc., 2009).

The model is a very straightforward implementation of the theory about matrix factorization:

$$\hat{R}_{ui} = \mu + b_i + b_u + Q_i^T P_u$$

Where μ is the standard deviation and b_i and b_u are set to zero in the first iteration.

Non-negative matrix factorization is a variant of SVD that only uses positive values in its factorization. In theory, this means that the model will use fewer values for their computations as it only cares about the positive ones. Thus, this can lead to faster computation time while retaining the accuracy of other matrix factorization models. The model has increased in popularity recently and is especially popular in the field of computer vision. According to its authors, the model is fast, easy to implement, and produces good results (Luo et al. 2014). The model is very similar to the general matrix factorization implementation.

$$\hat{R}_{ui} = P_u^T Q_i$$

To optimize the performance of the three methods, a Randomized Search CV (Cross-Validation) approach was used to tune the hyperparameters. This approach was chosen due to the time and resource constraints that limit trying all hyperparameter combinations exhaustively. Randomized Search CV is a technique that allows automatic exploration of parameters within a specified range of values. This process is performed randomly, selecting hyperparameter combinations from a predefined range. For each combination, the model is evaluated using cross-validation techniques to objectively measure its performance. A CV value of 5 was chosen as it is still computationally efficient compared to higher CV values, such as 10 or 20-fold. Furthermore, the models that have been built using the baseline, SVD, KNN, and NMF approaches will be compared to get the best performance.

One of the most common ways of evaluating recommendation models is by using some kind of error prediction. The most popular error metric is the Root Mean Square Error (RMSE) which predicts the accuracy of the predicted ratings for items recommended to users (Ricci 2011).

RMSE by the predicted rating and the actual rating is given by:

$$\sqrt{\frac{1}{|T|}\sum_{(u,i)\in T}(\hat{r}_{ui}-r_{ui})^2}$$

A lower RMSE value indicates that the model has a smaller error rate in predicting user ratings of items. Based on this, the following results are obtained.

	Model	CV Performance - RMSE	Model Condiguration
0	Baseline	0.871789	N/A
1	KNN Baseline	0.848675	{'k': 25, 'sim_options': {'name': 'pearson_bas...
2	SVD	0.865616	{'lr_all': 0.01, 'n_factors': 50, 'reg_all': 0...
3	NMF	0.869731	{'n_factors': 35, 'n_epochs': 90}

Figure 5. Model Comparison

KNN Baseline has the smallest RMSE value with hyperparameter k = 25 (refers to the number of nearest neighbors considered by the model, this number affects the model in understanding user preferences), pearson_baseline (calculation of similarity between items and users) and min_k = 2 (minimum value of nearest neighbors required to make predictions). Next, the accuracy of the model is tested on test data, and the results of RMSE tuning and RMSE test can be seen in the table below.

Model	RMSE-Tuning	RMSE-Test
User to User CF	0.848675	0.84056

Figure 6. RMSE Tuning and RMSE Test

The tuning RMSE with a value of 0.848 is used to select the best model during the hyperparameter tuning process. Meanwhile, the test RMSE with a value of 0.840 is used to measure the performance of the model on never-before-seen test data. This helps assess the extent to which the model can be used to provide accurate recommendations in real-world situations.

III. Conclusion

In developing a recommendation system, there are several approaches that can be used. First, there is non-personalized recommendation, which includes searching for top popular movies and filtering by genre. Next, there is a personalized recommendation approach, which uses content-based recommendations. However, since there are some limitations with this approach, a user-to-user collaborative filtering approach is used. The test results show that the KNN Baseline algorithm with hyperparameter k=25, using Pearson Baseline, and min_k=2, is the best model. This model has an RMSE result on the test of 0.840, which indicates a good level of accuracy in providing recommendations to users.

IV. Reference

Ashish Goel. 2009. MS&E 235: Lecture 7Long Tail. Notes by Rio Goodman. Lecture Notes in Cornell University.

Bell, R., Y. Koren, and C. Volinsky (2009). “Matrix Factorization Techniques for Recommender Systems”. In: Computer 42.8, pp. 30–37. issn: 0018- 9162.

Jahrer M., Toscher A. 2012. Collaborative Filtering Ensemble. Journal of Machine Learning Research 18:61–74, 2012.

Leskovec J. Rajaraman A. Ullman D. J. 2014. Mining of Massive Datasets.

Luo, X., M. Zhou, Y. Xia, and Q. Zhu (2014). “An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems”. In: IEEE Transactions on Industrial Informatics 10.2, pp. 1273–1284. issn: 1551-3203. doi: 10.1109/TII.2014.2308433.

Ricci, Francesco, Lior Rokach, and Bracha Shapira (2011). “Introduction to Recommender Systems Handbook”. In: Recommender Systems Handbook. Ed. by Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. Boston, MA: Springer US, pp. 1–35. isbn: 978-0-387-85819-7 978-0-387- 85820-3. doi: 10.1007/978-0-387-85820-3_1.

Schafer J. B., Frankowski D., Herlocker J., Sen S. 2007. Collaborative Filtering Recommender Systems.

Soucy P., Mineau G.W. 2005. Beyond TFIDF Weighting for Text Categorization in the Vector Space Model. Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005.

Torstensson. 2019. Comparing neighborhood and matrix factorization models in recommendation systems.