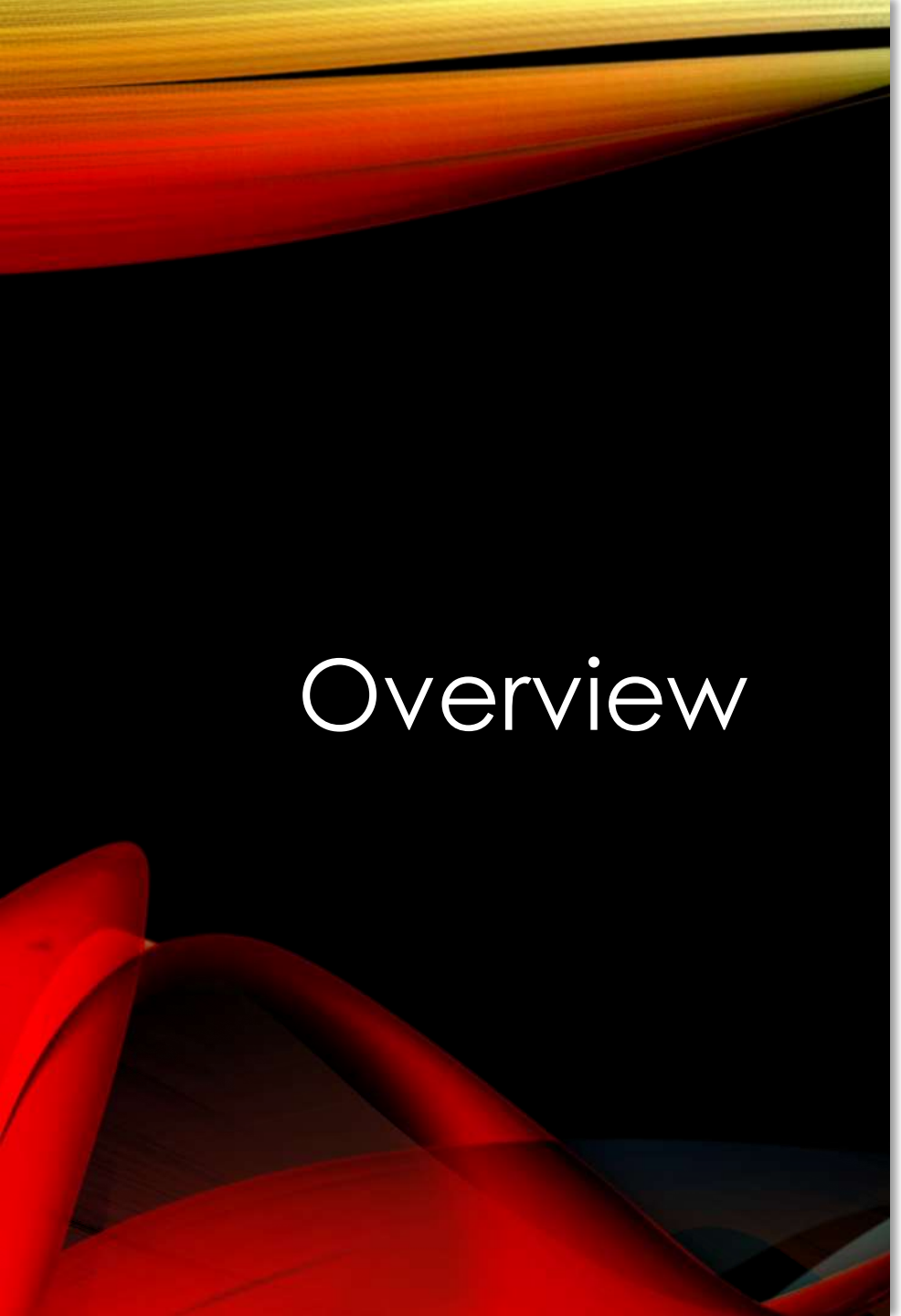
A Comparative Study in Movie Recommendation Systems

Comparing Matrix Factorization and KNN-based recommendation systems.

Team: Sharanya Banerjee, Radhika Kulkarni, Pooja Bihani, Hari Murugan Ravindran, Arushi Gaur and Swarali Chine



Overview

- Understanding and designing the model: Matrix Factorization
- Understanding and designing the model: KNN-based recommendation
- Comparing outcomes from both models
- Generating inference from the comparison

Problem Description

- The entertainment industry today is massively driven by Movies. Ever since the emergence of on-demand platforms like Netflix, Prime-Video, etc. the need to leverage data and technology to streamline the process of content consumption has increased manifold.
- With the advent of Machine Learning, these platforms can accurately recommend movies to watch based on our watching habits.
- These models use clustering techniques to identify data points to make accurate predictions.
- In this project, we aim to compare such models using *Collaborative Filtering and Matrix Factorization* and *Item-based KNN,* and test it on the 20M MovieLens dataset to predict users' movies to watch.

Introduction to Recommendation Systems

- Implemented by **inverse indexing**; An **information retrieval** method where **it takes a feature and maps it to those vectors that contain this feature** (Subset of Information Theory)

❑ **Content-Based Systems** (Ex. KNN Based Recommendation)
❑ **Collaborative Filtering Systems** (Ex. Matrix Factorization)
❑ **Hybrid Systems**

- ***Content-Filtering****:* This system **uses the features of a particular item** to recommend other similar items.

  For ex: If a user looks up for solid-colored t-shirts, the system might recommend t-shirts or solid-colored sweatshirts

- ***Collaborative-Filtering:*** systems **use the actions of a user to recommend other items**. In general, there can be **user-based** and **item-based** collaborative filtering systems

  1. *User-based systems:* It uses the **patterns of users like another user** to recommend products
  2. *Item-based systems:* It uses the **patterns of users who browsed the same item** as another similar user to recommend products

Matrix Factorization based Recommendation

Matrix factorization is a class of **collaborative filtering algorithms** used in recommender systems

They work by **decomposing** the **user-item interaction matrix into** the product of **two lower dimension rectangular matrices**

The **user-item interaction** matrix is decomposed into the product of two lower dimensionality rectangular matrices by **matrix factorization algorithms**

Users and items are the two main components of a recommender system - **Assume** there are **m users and n items**

The **purpose** of our recommendation system is to **create a m x n matrix** (also known as the **utility matrix**) that **contains** each **user-item pair's rating (or preference)**

Cold Start Problem: Because we only have ratings for a **small number of user-item pairs**, this matrix is **usually relatively sparse at first**

Dimensionality Reduction is done using **Singular Value Decomposition (SVD)**

$$\hat{y}_{ij} = u_i \cdot v_j$$

KNN-based Recommendation

k-Nearest Neighbors (kNN), is the optimization problem of **finding points in each set that are closest to a given point**

KNN becomes impractical in most modern-day applications, which have **massive datasets with high dimensionality**, as it requires an **exhaustive search**

CURSE OF DIMENSIONALITY: As the number of data points and number of dimensions increase, all the points start being equidistant from each other

We **need** to **trade-off some level of accuracy for a substantial improvement in speed**

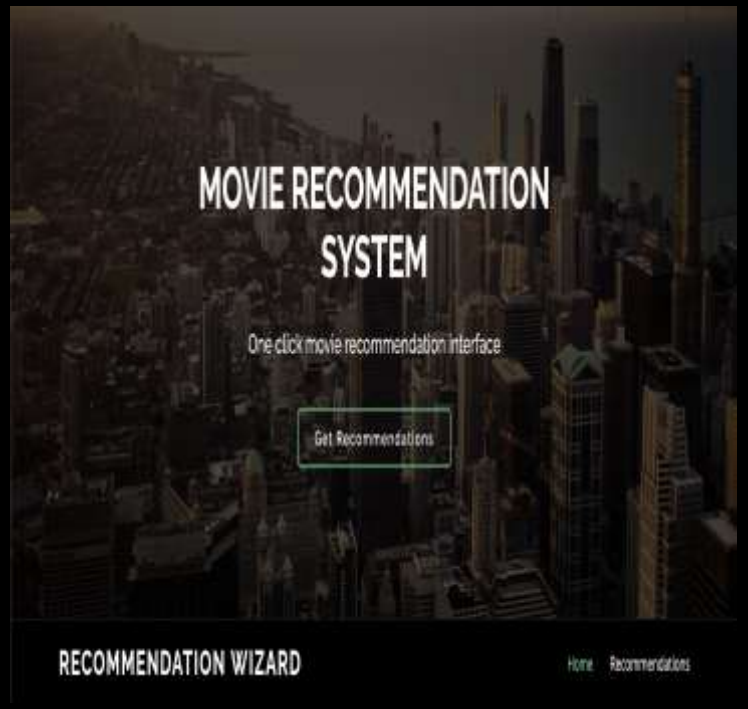
Key Task: Train **distance function between user and item vectors** for item-based movie suggestions (Deep Learning)

Dataset Description

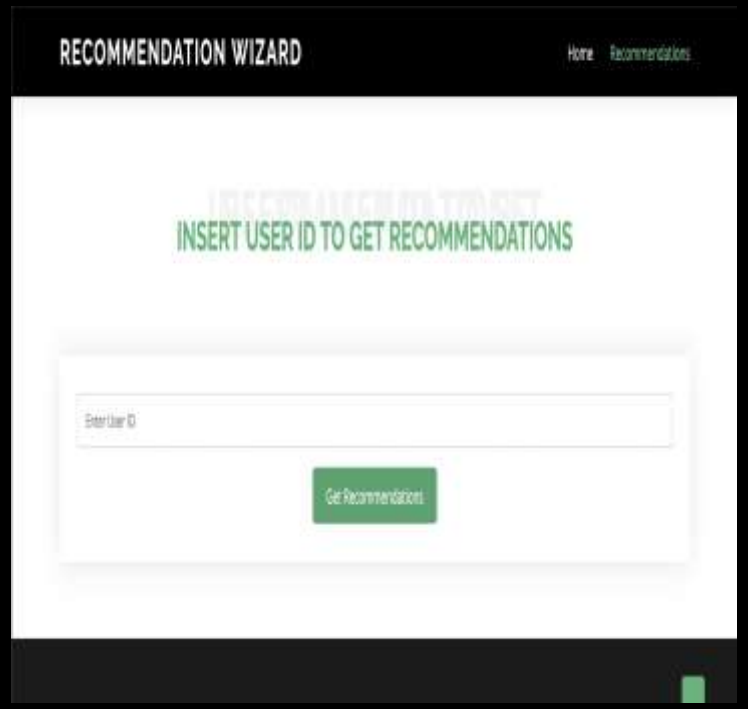
- Dataset selected: MovieLens 20M Dataset; Link: https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset
- The data is arranged in six files.
- It contains **20000263 ratings and 465564 tag applications** across **27278 movies**. This data was created by **138493 users** between **January 09, 1995**, and **March 31, 2015**. All selected users had rated at least **20** movies.
- The **tag.csv** mainly contains tags suitable for movies by Users. The columns in tag.csv are **userId, movieId, tag and timestamp.**
- The **rating.csv** contains ratings for movies by Users. The ratings range from 0.5 to 5 with 0.5 increments. The columns in rating.csv are **userId, movieId, rating and timestamp.**
- The **movie.csv** mainly contains movie titles with suitable genres for those movies. The columns in movie.csv are **movieId, title and genres.**
- The **link.csv** contains identifiers in an external movie database like IMDB. The columns in link.csv are **movieId, imdbId and tmbdId.**
- The **genome_scores.csv** contains movie-tags relevance data. The columns in genome_scores.csv are **movieId,tagId and relevance.**
- The **genome_tags.csv** contains tag descriptions. The columns in genome_tags.csv are **tagId and tag.**

UI Walkthrough

Homepage



Recommendation



Outcome

Evaluation Metrics

Mean Absolute error

- It is the difference between the actual value(rating) and the predicted value.

$$MAE = \underbrace{\frac{1}{n}}_{\text{test set}} \sum_{i=1}^{n} \underbrace{|y_i}_{\text{predicted vaue}} - \underbrace{\hat{y}_i|}_{\text{actual value}}$$

Root Mean Squared Error

- RMSE is the standard deviation of the residuals (prediction errors).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Recall and Precision

- The ability of a model to find all the relevant cases within a data set.
- The ability of a classification model to identify only the relevant data points.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Outcome

```
i = 1
for trainset, testset in kf.split(data):
    print("Split:", i)
    predictions = algo.fit(trainset).test(testset)
    accuracy.rmse(predictions, verbose=True)
    precisions, recalls = precision_recall_at_k(predictions, k=5, threshold=4)

    print("Precision:", sum(prec for prec in precisions.values()) / len(precisions))
    print("Recall:", sum(rec for rec in recalls.values()) / len(recalls))
    i+=1
```

```
Split: 1
RMSE: 0.8640
Precision: 0.8699453551912583
Recall: 0.23114313901461234
Split: 2
RMSE: 0.8762
Precision: 0.8660382513661213
Recall: 0.21820739170821452
Split: 3
RMSE: 0.8654
Precision: 0.8726775956284165
Recall: 0.22114770490196428
Split: 4
RMSE: 0.8646
Precision: 0.8518032786885261
Recall: 0.21860234416573648
```

Comparison of Outcomes

PARAMETERS	MATRIX FACTORIZATION	K-NEAREST NEIGHBOURS
Mean Absolute Error	0.7460	0.7677
Root Mean Squared Error	0.8640	0.8762
Precision and Recall	Precision: 0.869453551912583 Recall: 0.2311431390146123 4	Precision: 0.8660382513661213 Recall: 0.2182073917082145 2

Inference

- From the above table we can see minor differences in Metric Values between the two approaches to recommendations.
- Matrix Factorization gives us **lower Error values** compared to KNN based recommendation.
- Also, Matrix Factorization has a **higher precision and recall** compared to KNN based recommendation.
- This helps us to infer that Matrix Factorization performs a better job in returning the expected outcomes when compared to the latter.
- Hence, we can argue that Matrix Factorization is **better suited** for recommendation applications.

Project Timeline

Task	Description	Deadline
Research and gather related papers	Research on existing algorithms, make plans for the project, identify the sub-tasks.	10th March
Analyzing existing systems	Identify algorithms that can have best performance/accuracy.	13th March
Preprocessing the datasets	Understand, clean and preprocess the dataset.	15th March
Designing the Systems	Gathering the requirements and designing the system	17th march
Implementation	Coding and testing the program.	21st March
Comparison and Inference	Compare based on metrics from different models	23rd March
Project Presentation	Make the presentation powerpoint	4th April
Final code changes	Finish coding and testing, prepare for demo.	15th April
Demo	Prepare for demo and present it	25th April
Project Report	Write the final project report.	2nd May

References

- https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-1-knn-item-based-collaborative-filtering-637969614ea
- https://www.analyticsvidhya.com/blog/2021/05/item-based-collaborative-filtering-build-your-own-recommender-system/
- https://analyticsindiamag.com/collaborative-filtering-vs-content-based-filtering-for-recommender-systems/
- https://en.wikipedia.org/wiki/Cold_start_(recommender_systems)
- https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset

Code (Github Link)

- https://github.com/RadhikaKulk/CSE573-SWM-Movie-Recommender