Movie Recommendation System

Group 01

Aditi Shashank Joshi - 1222838916 - ajoshi64@asu.edu
Harshitha Verma - 1223176266 - hverma7@asu.edu
Kaushik Hegde Kota - 1219807888 - kkota2@asu.edu
Kavya Bathini - 1223231230 - kbathini@asu.edu
Mamatha Juluru - 1219438753 - mjuluru@asu.edu
Vishrut Kumar Jha - 1221914707 - vkjha@asu.edu

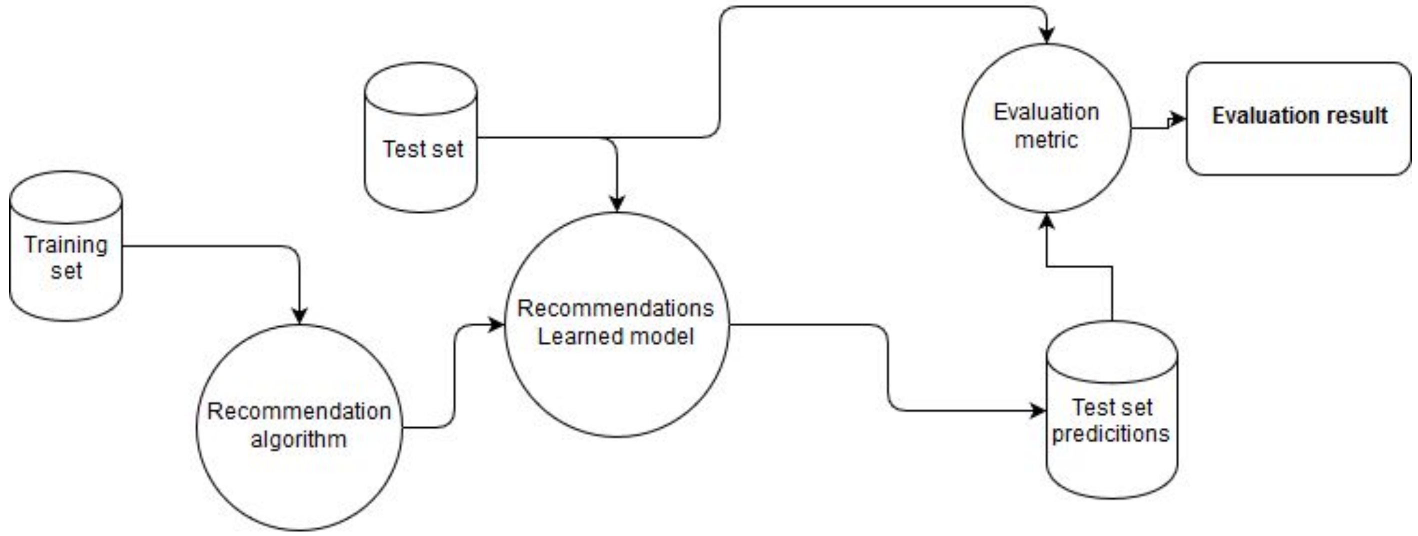
Problem Definition

There is a need for a recommender system that can tailor the platform based on the user's tastes, whether it be for streaming movies, making purchases online, or social networking because there are so many options available nowadays.

In essence, a movie recommendation engine tracks our viewing habits and generates suggestions using pertinent information. This not only makes it simpler for people to choose their favorite films, but it also benefits businesses by allowing them to identify the genres of stories that have the broadest appeal to customers in order to boost sales.

The recommendation system in this project will be built using the K-Nearest-Neighbor algorithm based on cosine similarity. The collaborative filtering approach will suggest movies based on the preferences of the other user. The Matrix Factorization will serve as the foundation for our recommendation engine.

System Architecture



Dataset Description

We will be using the data set MovieLens

- Consists of 25 million ratings and one million tag applications
- Includes tag genome data with 15 million relevance scores across 1,129 tags.
- The users had rated at least 20 movies and each user was represented by an id.
- Ratings are in the range [0.5, 5]
- The description of the data in the dataset are contained in six files - tag.csv, rating.csv, movie.csv, link.csv, genome_scores.csv, genome_tags.csv.

We will be using the data set The Movies Database TMDB 5000

- Contains movies and credits csv files
- Contains 4803 rows in each csv. Movies.csv (4803,20) , Credits.csv (4803,4)
- Movies.csv contains metadata related to each movie such as the genre, language, tagline, and keywords.
- Credits.csv contains the movie id, title, cast and crew details

Research Plan

- Movie recommendation system has two main elements: **users** and **items**, the movie recommendations are done for the users and the movies are items.
- There are multiple datasets available which can be used, we thoroughly studied the methodology and considered the **MovieLens 25M** dataset and the **TMDB 5000** dataset.
- We have built a content based recommendation system using NearestNeighbors algorithm.
- We also felt the need to build a recommender system using **Collaborative Filtering** as it is free of domain knowledge because it is learned based on other user’s tastes.
- We are planning to create a recommendation system using the following algorithms:
    - Matrix Factorization based recommendation
    - KNN Based Recommendation

State-of-Art Methods & Algorithms

Content-Based Filtering

- Extracts the features to suggest a movie to the user
- It doesn’t need data from other users’ to develop recommendations and are tailored specific for the user
- Feature matrix is created to calculate the frequency, weights and influence of a particular feature in the recommendation system.
- Cosine similarity is used to generate similarity scores of the features
- NearestNeighbors is used to predict the rating of the movie and suggestions for the user

State-of-Art Methods & Algorithms

Collaborative Filtering

- Suggests recommendations based on similar users' watchlist and other attributes
- Uses 2 ratings - implicit and explicit
- Uses a similarity matrix factorization method - finds similarity between users and items using various methods, such as cosine similarity, Pearson similarity.
- Also Uses K-nn to provide recommendations
- Whenever a new user joins, a correlation vector consisting of the user and items similarities is developed

Evaluation Metrics

For evaluation purposes, we are making a test-train split of 80:20 to the dataset. We employ the following baseline models:

- Matrix Factorization: A collaborative filtering method, to find the relationship between items and defined entities. Using this, we identify the latent features, user associations and movie matrices, to find similarity and make predictions on both, items and entities.
- Item-based KNN: This relies on item-feature similarity. The data-points are spread across several clusters to make inference for new samples.

Then, we evaluate the model based on the following metrics:

- Mean Absolute Error
- RMS (Root Mean Squared) Error
- Precision/Recall
- Accuracy
- F1 Score

Evaluation Metrics

Precision/Recall -> Translate to Movie Recommendation

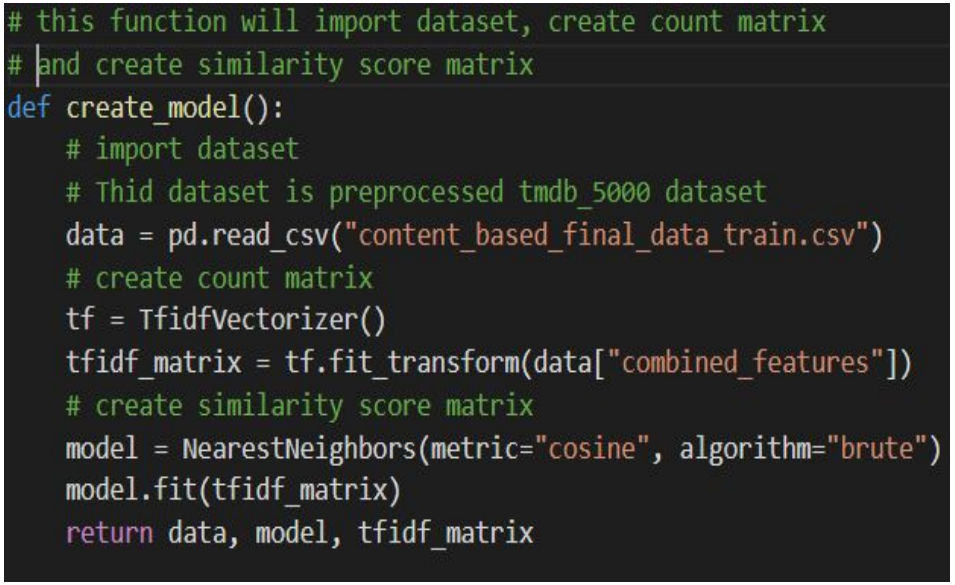
Given an assumed rating for example 3.5

- Relevant items are already known in the data set
  - Relevant item: Has a True/Actual rating >= 3.5
  - Irrelevant item: Has a True/Actual rating < 3.5
- Recommended items are generated by recommendation algorithm
  - Recommended item: has a predicted rating >= 3.5
  - Not recommended item: Has a predicted rating < 3.5

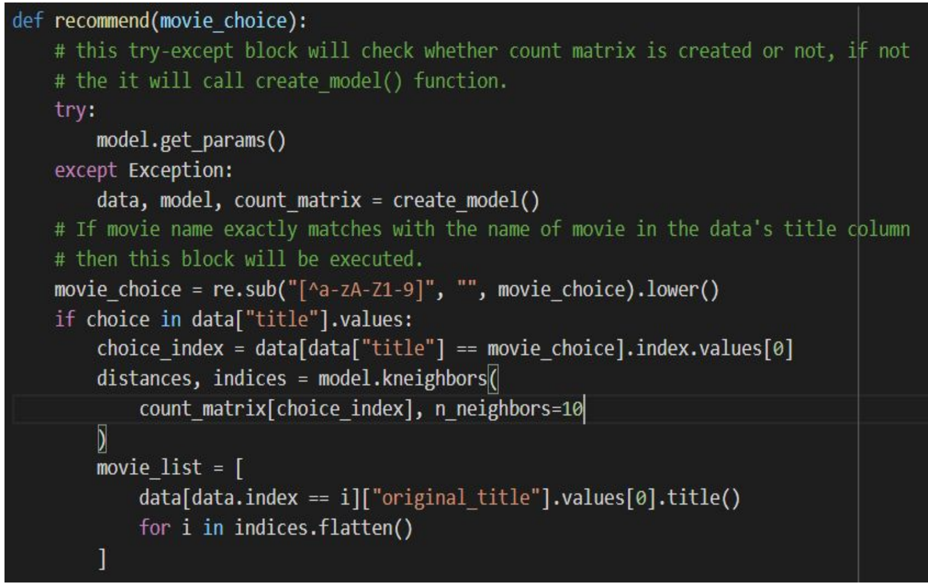
Precision = (# of recommended items that are relevant) / (# of recommended items)
Recall = (# of recommended items that are relevant) / (total # of relevant items)

Evaluations - Preliminary Findings

Content Based Recommender

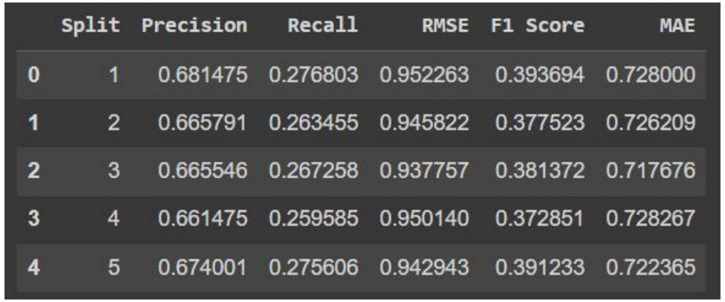


```python
# this function will import dataset, create count matrix
# and create similarity score matrix
def create_model():
    # import dataset
    # Thid dataset is preprocessed tmdb_5000 dataset
    data = pd.read_csv("content_based_final_data_train.csv")
    # create count matrix
    tf = TfidfVectorizer()
    tfidf_matrix = tf.fit_transform(data["combined_features"])
    # create similarity score matrix
    model = NearestNeighbors(metric="cosine", algorithm="brute")
    model.fit(tfidf_matrix)
    return data, model, tfidf_matrix
```



```python
def recommend(movie_choice):
    # this try-except block will check whether count matrix is created or not, if not
    # the it will call create_model() function.
    try:
        model.get_params()
    except Exception:
        data, model, count_matrix = create_model()
    # If movie name exactly matches with the name of movie in the data's title column
    # then this block will be executed.
    movie_choice = re.sub("[^a-zA-Z1-9]", "", movie_choice).lower()
    if choice in data["title"].values:
        choice_index = data[data["title"] == movie_choice].index.values[0]
        distances, indices = model.kneighbors(
            count_matrix[choice_index], n_neighbors=10
        )
        movie_list = [
            data[data.index == i]["original_title"].values[0].title()
            for i in indices.flatten()
        ]
```

Evaluations -Preliminary Findings



	Split	Precision	Recall	RMSE	F1 Score	MAE
0	1	0.681475	0.276803	0.952263	0.393694	0.728000
1	2	0.665791	0.263455	0.945822	0.377523	0.726209
2	3	0.665546	0.267258	0.937757	0.381372	0.717676
3	4	0.661475	0.259585	0.950140	0.372851	0.728267
4	5	0.674001	0.275606	0.942943	0.391233	0.722365

KNNBase Collaborative Filtering

- The ratings are on the scale of 0.5 to 5.
- KNNBase model from surprise on the complete dataset
- by using kfold dataset splits to overcome overfitting.
- Used build_anti_testset create a complement of the train dataset.
- This will generate the entries for the movies which the user has not rated.
- Model will be used to predict ratings for the movies in the anti_testset
- Uses the cosine or the pearson similarity to generate the predicted ratings.
- Sort the movies for each user based on the ratings and present the top n recommended movies

Evaluations -Preliminary Findings

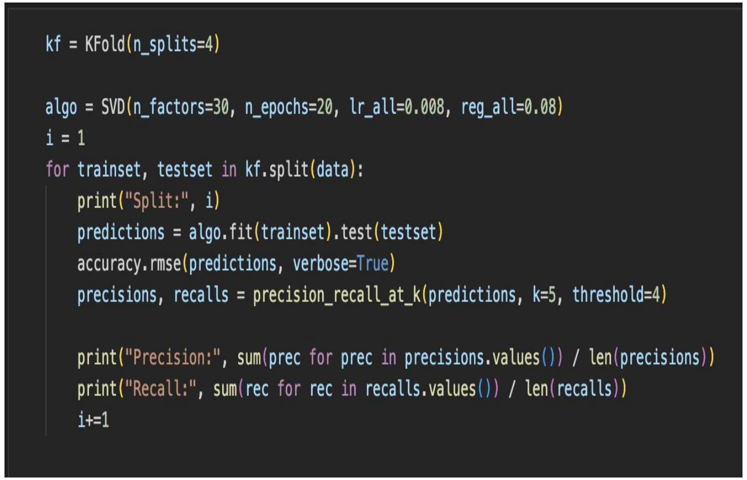
Matrix Factorisation Collaborative Filtering

- The ratings are on the scale of 0.5 to 5.
- SVD model from surprise on the complete dataset
- By using kfold dataset splits to overcome overfitting.
- Used build anti_testset create a complement of the train dataset.
- This will generate the entries for the movies which the user has not rated.
- Model will be used to predict ratings for the movies in the anti_testset
- Uses the Mean Squared Difference, cosine or the pearson similarity to generate the predicted ratings.
- Sort the movies for each user based on the ratings and present the top n recommended movies

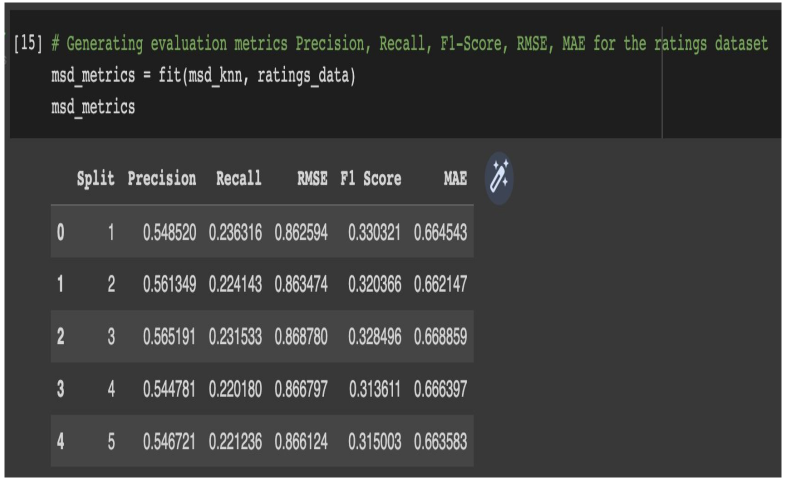
Evaluations - Preliminary Findings

Matrix Factorisation

We have trained the Singular Value Decomposition model from surprise on the complete dataset by using kfold dataset splits to overcome overfitting. For each split, we determine the Precision, recall and RMSE values. Top N movie IDs that are recommended by the recommendation system:



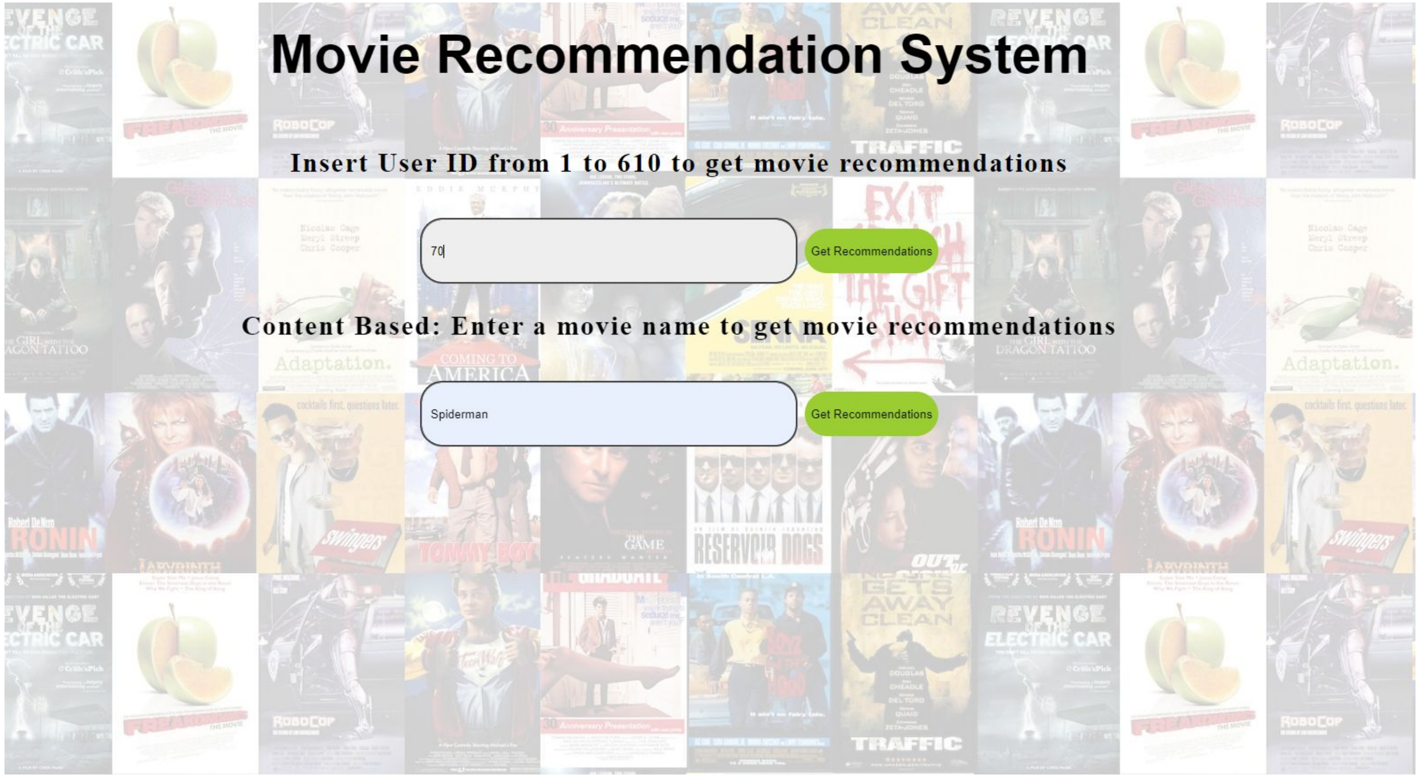
```python
kf = KFold(n_splits=4)

algo = SVD(n_factors=30, n_epochs=20, lr_all=0.008, reg_all=0.08)
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
```



```python
[15] # Generating evaluation metrics Precision, Recall, F1-Score, RMSE, MAE for the ratings dataset
     msd_metrics = fit(msd_knn, ratings_data)
     msd_metrics
```

```
   Split  Precision    Recall      RMSE  F1 Score       MAE
0      1   0.548520  0.236316  0.862594  0.330321  0.664543
1      2   0.561349  0.224143  0.863474  0.320366  0.662147
2      3   0.565191  0.231533  0.868780  0.328496  0.668859
3      4   0.544781  0.220180  0.866797  0.313611  0.666397
4      5   0.546721  0.221236  0.866124  0.315003  0.663583
```

UI Interface - HomePage



Movie Recommendation System

Insert User ID from 1 to 610 to get movie recommendations

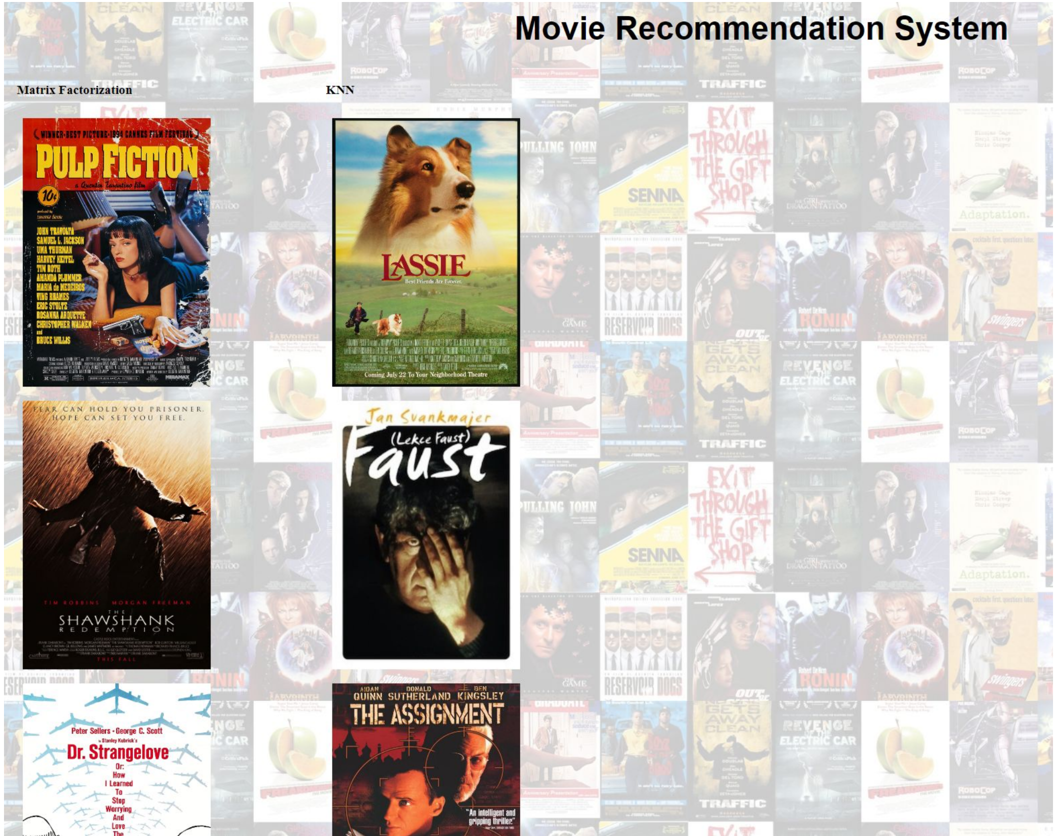
70

Content Based: Enter a movie name to get movie recommendations

Spiderman

UI Interface - Results

Collaborative FIltering:

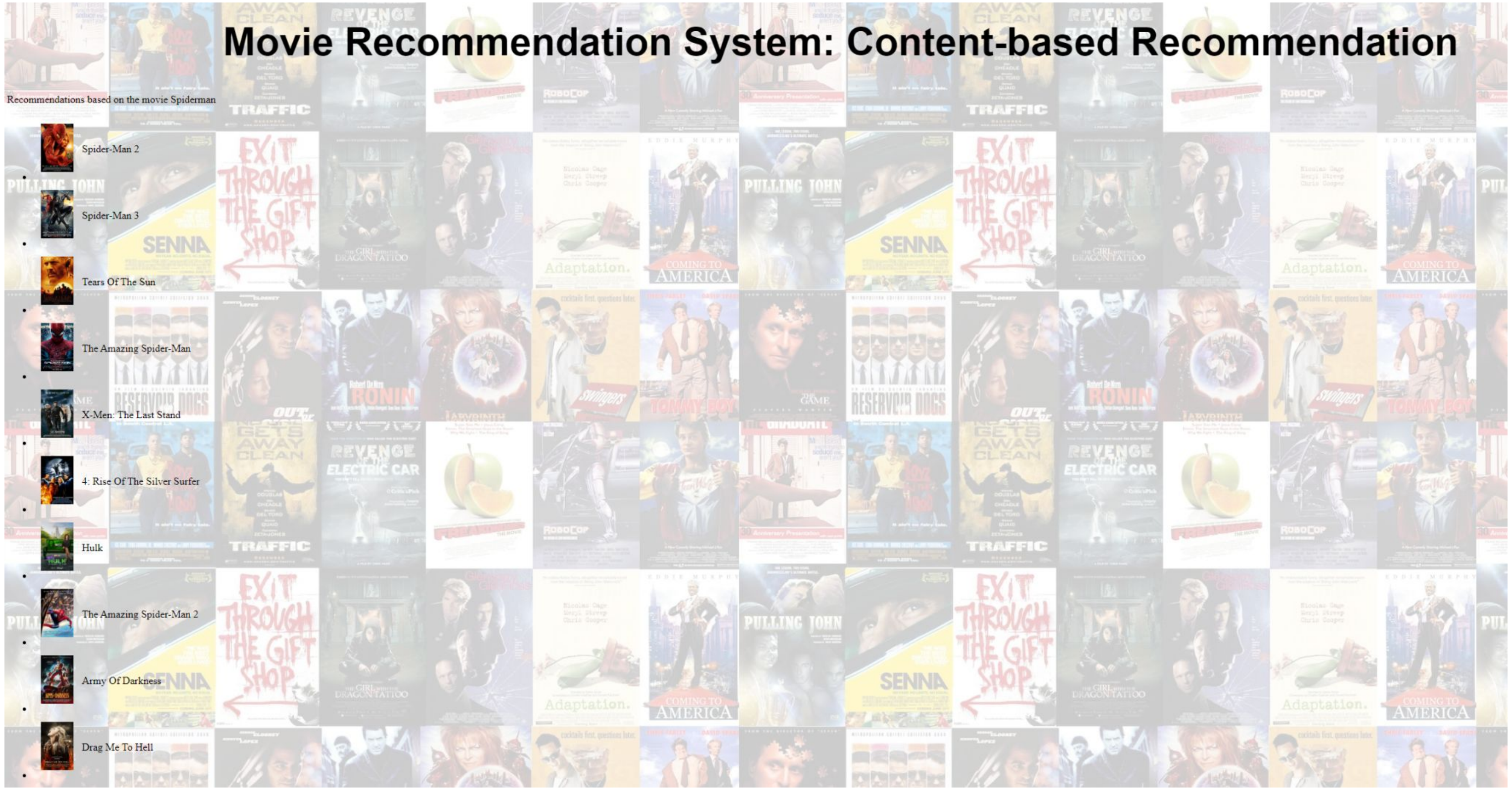


Movie Recommendation System

Matrix Factorization

KNN

UI Interface - Results



Movie Recommendation System: Content-based Recommendation

Recommendations based on the movie Spiderman

- Spider-Man 2
- Spider-Man 3
- Tears Of The Sun
- The Amazing Spider-Man
- X-Men: The Last Stand
- 4: Rise Of The Silver Surfer
- Hulk
- The Amazing Spider-Man 2
- Army Of Darkness
- Drag Me To Hell

Project Plan

Task	Description	Deadline
Initial Research - Papers, Blogs	Conduct Initial research for the problem statement and know the current State of the Art methods used	Done
Understand and Prepare the Dataset	Understand the dataset and prepare the dataset for modeling through data preprocessing.	Done
Algorithms Survey - Evaluate and Finalize	Survey various STOA algorithms and check performance on small blocks of the 25M movie dataset.	Done
Modeling - Implementation	Implement models on the entire dataset and check model performance	Done
Project Presentation	Create a presentation to explain the problem station and proposed solutions.	Done
Demo of the Project	Present the working model and the created presentations	11/16
Final Project Report	Summarize and Record the Project details in a Project Report	12/2

References

1. Schafer, Ben & J, Ben & Frankowski, Dan & Dan, & Herlocker, & Jon, & Shilad, & Sen, Shilad. (2007). Collaborative Filtering Recommender Systems.
2. Evaluation Metrics - https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2
3. Grouplens, “25M MovieLens Dataset”, Available at - https://grouplens.org/datasets/movielens/25m/
4. Kaggle, “25M MovieLens Dataset” Available at - https://www.kaggle.com/datasets/garymk/movielens-25m-dataset
5. Analytics Vidhya, "Movie recommendation system using KNNs", Available at - https://www.analyticsvidhya.com/blog/2020/08/recommendation-system-k-nearest-neighbors/
6. Tahsin Mayeesha - Medium, “Anime recommendation using KNNs”, Available at https://medium.com/learning-machine-learning/recommending-animes-using-nearest-neighbors-61320a1a5934
7. Kaggle, “Movie Recommendation system for Deployment”, Available at https://www.kaggle.com/code/terminate9298/movie-recommendation-system-for-deployment/notebook
8. Towards Data Science, “Matrix Factorization, Recommender Systems”, Available at https://towardsdatascience.com/recommendation-system-matrix-factorization-d61978660b4b
9. Wiki, “Matrix Factorization for Recommender Systems”, Available at https://en.wikipedia.org/wiki/Matrix_factorization_(recommender_systems)

Github Repository Link

https://github.com/aadiiitiii/CSE573-Movie-Recommendation

Thank you!