TEAM F-FINAL PROJECT ON RECOMMENDATION SYSTEMS:REPORT

Ιωαννίδης Κωνσταντίνος 2931,Μάρης Αθανάσιος 3045,Τσατάνης Παναγιώτης 3014

January 25, 2024

Abstract

In the last 20 years,recommender systems have attracted a lot of interest due to the explosion in the amount of data in online applications.In everyday life, people often refer to the opinions and advice of others when it comes to purchasing for a service from a company, buying an item in a marketplace or a store, watching a movie at a cinema, etc.We observe that the ability to recommend items that a person may like,is an important factor in their lives,given the many options they have.Additionally,recommender systems aid the finance sector,because it is more possible for consumers to purchase items and services that they really like.For the above reasons,we decided to delve deeper into the art of recommender systems and to learn different techniques that are used to implement the above systems.

Some of the techniques we discussed and analyzed are Collaborative-Filtering,Content-Based Recommending and Hybrid Approaching.In Collaborative-Filtering,a user is recommended items based on the past ratings of all users collectively,while in Content-Based Recommending,the user is recommended items that are similar in content to items the user has liked in the past, or matched to attributes of the user.In a hybrid approach,both of the above methods are combined.We decided to implement a recommender system in 7 different ways.The techniques we used was SVD(Singular Value Decomposition),NMF(Non-Negative Matrix Factorization),SGD(Stochastic Gradient Descent),k-NN(Nearest Neighbors),NN with Keras(Neural Network),GPT-like recommendations and one method we invented,the alphas method.So,in our paper,we will present our literature,aka the papers we based our project on,the dataset we used,the methods that we implemented,as well as the metrics and the experiments we did,in our path to produce our results.

1 Introduction

Nowadays,we have an abundance of items,products and services available on the internet.So,users,and customers in general,are having difficulty filtering all those options and discovering items that align with their preferences.Recommender systems solve this problem by analyzing user behavior, preferences, and historical interactions to provide personalized recommendations.Also, recommender systems go beyond presenting familiar items and also aim to introduce users to new and relevant items they may not have discovered on their own.

Our motivation for implementing this project,is something very familiar with the everyday user.We all have Netflix and we were curious as to how it manages to recommend users,including us,new movies and shows.Also,recommender systems represent an active and rapidly evolving field within computer science and information technology.So,another reason for studying recommender systems is because the knowledge we will acquire,will be beneficial for our careers in the future.Additionally,many businesses leverage recommender systems to boost sales, increase user engagement, and enhance customer satisfaction.So our ability to solve business challenges will increase,therefore opening more doors and opportunities into the business sector.

We used the MovieLens dataset in our experiment.Regarding the input,we used the columns UserId,MovieId,Title,Genres and Rating.We are going to get into more detail for the input of each method when we will analyze each method at a later stage.Generally,our output is the predicted movies and the rating of the predicted movies.But we will be more specific later.

2 Literature Review

In our literature review we present the short summary of 5 papers,which we occasionally consulted during the execution of our project.

Machine Learning Algorithms for Recommender System - a comparative analysis:

The implementation of the system can be performed by various techniques. In this paper, we have discussed Content Based Filtering, Collaborative Filtering[10], Hybrid Content-Collaborative Based Filtering, k-mean clustering Based and Naive-Bayes Classifier based techniques.
Also,the cold-start problem is one of the most commonly encountered challenges of the recommendation system. It is also known as the new user problem as it creates problem of generating recommendations for the new user.
The algorithms used in this paper are K-means clustering, collaborative filtering, content based filtering, hybrid filtering and naive Bayes
In order to measure the precision of their algorithms,the authors convert some of their rated movies to unrated movies.

A New Algorithm for Solving Data Sparsity problem Based-On Non Negative Matrix Factorization in Recommender Systems:

The algorithms that the authors use are SVD and NMF.
One of the advantages of NMF algorithm in comparison to SVD is that NMF method gives basis and weight vectors under non negative constraints and offers a better understanding and interpretation of the data. While the singular value decomposition can lead to lack of proper interpretation of the non negative data in terms of physical reality and intuitive.
Because their ratings matrix in the NMF algorithm is sparse,the authors dont use 0 to fill in their missing values.What they were using is the mean rating by user/movie/total.This helps the NMF algorithm.
As an observation,they saw that NMF is problematic in big dimensions,while SVD isn't.

Improving Accuracy of Recommender System by Clustering Items Based on Stability of User Similarity

The authors perform k-NN using cosine similarity and Pearson similarity.First,they separate the movies according to genre and then they perform k-NN in every genre.Then,they seperate the movies one more time,this time according to user similarity (Vector cosine knn).That is very clever because metadata are produced by users,so they carry error.
Their approach is totally different from existing approaches, in the sense that they do not use item-to-item distance. Instead, they use item-to-item-group distance.

Recommender Systems: Collaborative filtering (CF) and content-based filtering (CBF) are two prominent methodologies in the domain of recommender systems, with hybrid techniques aiming to merge their functionalities. CF systems analyze historical interactions among users to recommend items, while CBF systems rely on profile attributes. Hybrid approaches seek to combine the strengths of both. The architecture and evaluation of recommender systems remain active research areas.

CF, characterized by analyzing usage data across users to find well-matched user-item pairs, contrasts with the older content filtering methodology, rooted in information retrieval. Content filtering does not explicitly utilize information across the entire user base for recommendations. Early successes of CF include the GroupLens system. Matrix factorization techniques, rooted in numerical linear algebra and statistical matrix analysis, have emerged as state-of-the-art in CF.

Neighborhood-based CF assigns weights to users based on similarity with the active user, selecting a neighborhood of similar users to compute predictions. Item-based CF matches a user's rated items to similar items, leading to faster online systems and improved recommendations.

Significance weighting and default voting address challenges in CF, such as correlations based on few co-rated items. Inverse user frequency adjusts ratings based on item popularity.

Model-based techniques, including latent factor and matrix factorization models, estimate parameters of statistical models for user ratings. These models assume a hidden lower-dimensional structure in the data, leading to more accurate recommendations.

Content-based recommending compares item descriptions with user interests, often using cosine similarity or classification algorithms. Hybrid approaches merge content-based and collaborative filtering methods, either by combining separate ranked lists of recommendations or by using content-based predictions to augment CF.

Evaluation metrics for recommender systems include mean absolute error, root mean squared error, precision, recall, and area under the ROC curve. Challenges include sparsity in user ratings matrices and the cold-start problem for new items and users. Fraudulent behavior also poses a threat to the integrity of recommender systems, as they are increasingly used by commercial websites.

Latent factor models, a state-of-the-art methodology in model-based CF, assume a low-dimensional representation of users and items where user-item affinity can be accurately modeled. Matrix factorization techniques aim to find weighted low-rank approximations to the user-item matrix, considering factors such as loss functions and regularization terms.

In summary, recommender systems continue to evolve, with ongoing research focusing on improving accuracy, scalability, and robustness to various challenges in recommendation tasks. Hybrid approaches and advanced modeling techniques show promise in addressing these issues and enhancing user experience in diverse application

domains.

A Comprehensive Review on Non-Neural Networks Collaborative Filtering Recommendation Systems:

The authors discuss Memory-Based Collaborative Filtering.Memory-based algorithms compute predictions using the entire user-item interactions directly on the basis of similarity measures. They can be divided into two categories, User-based CF and Item-based CF.They then go into similarity computation,which is also divided into 2 categories,Pearson Correlation and Cosine-based similarity.

Also,they delve into Prediction and Recommendation.The way they make their predictions is through the formula of the weighted sum.Recommendation is done through the Top-N Recommendation.The Top-N recommendation algorithm aims to return the list of the most relevant items for a particular user. To recommend items that will be of interest to a given user, the algorithm determines the Top-N items that the user would like. The computation of these Top-N recommendations depends on whether the recommender system is a User-based CF or an Item-based CF.The Top-N recommendation can be further divided into User-based Top-N recommendation and Item-based Top-N recommendation.Despite their popularity, User-based recommendation systems have a number of limitations related to data sparsity, scalability and real-time performance,while Item-based algorithms are suitable to make recommendations to users who have purchased a set of items.The key advantages of Item-based over User-based collaborative filtering are scalability and performance.Additionally,they discuss the limitation of Memory-based CF.

The article moves on with Model-Based Collaborative Filtering.Instead of computing recommendations on the entire user-item interactions database, a model-based approach builds a generic model from these interactions, which can be used each time a recommendation is needed.The algorithms discussed by the authors are Singular Value Decomposition (SVD),Matrix Factorization (MF),Probabilistic Matrix Factorization (PMF),Non-negative Matrix Factorization (NMF) and Explainable Matrix Factorization (EMF).After discussing the dimensionality reduction techniques thoroughly,the authors present an overview of all those models.

Evaluation of metrics comes next.During the two last decades, dozens of metrics have been proposed in order to evaluate the behavior of recommender systems, from how accurate the system is to how satisfactory recommendations could be. Evaluation metrics can be categorized in four main groups, (a) prediction metrics, such as accuracy (b) set of recommendation metrics, such as precision and recall (c) rank of recommendations metrics like half-life and (d) diversity and novelty.Prediction accuracy measures the difference between the rating the system predicts and the real rating.Quality of set of recommendations is appropriate when a small number of observed ratings is known, and it is used to evaluate the relevance of a set of recommended items. After briefly dealing with quality of list of recommendations and novelty and diversity,the authors end this section with an overview of the evaluation metrics presented.

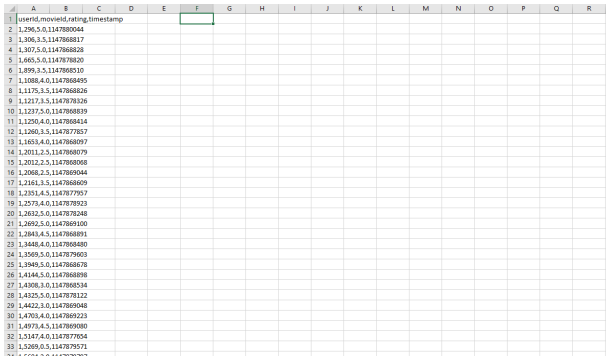
The dataset they used was MovieLens Dataset,the same as ours.They present the experimental results regarding the performance of the models and discuss the importance of ratings normalization.

3 Dataset and Features

For the project we used the MovieLens dataset.The MovieLens dataset contains 4 basic excel files.Those are links.csv,tags.csv,ratings.csv and movies.csv.From those 4,we worked mostly with ratings.csv and movies.csv.The ratings.csv contains 25 million ratings.That number is huge,so Google Colab couldn't handle it.So,for most of our experiments,we used a subset of that dataset,around 10000 ratings.We saw that 10000 was a good number regarding the finishing time of our experiments and when we started using more ratings,execution times were skyrocketing.

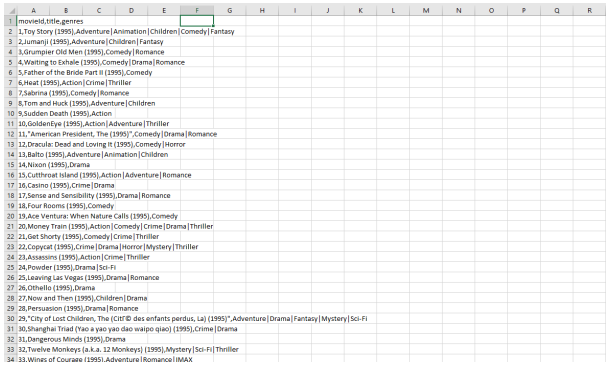
We started the preprocessing of the datasets by handling the missing values of our datasets.We substituted the missing values with 0.We then produced a heatmap of the correlation matrix,to depict the relation that genres have between them.We then print some statistical information regarding the dataset for each column,such as count,mean,standard deviation,min,1st 2nd and 3rd quartile and the max.After that,we made a function to remove outliers from a given dataset,and which we used for the Keras Implementation.The function is based on the IQR method.Following this function,we do some preliminary data manipulation with our dataframes,in order to "warm up".We start by sorting the ratings dataframe by movieId and by finding the average rating for each movie.We then find the number of votes for each movie and sort the movies according to number of votes.Also,we merged average rating and number of votes for each movie.As the last part of our preprocessing,we generated the ratings(user-movie) matrix,a matrix that is necessary for many of the following methods(SVD,NMF,etc.).We performed normalization of the data,by using a StandardScaler,in the Keras implementation.Data augmentation wasn't necessary as our dataset was big enough to conduct our experiments.We separated the dataset in train-test portions with a ratio of 80(train) - 20(test) in the Keras,GPT and Alpha methods.

Here is a glimpse of our ratings.csv dataset:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	userId,movieId,rating,timestamp																	
2	1,296,5.0,1147880044																	
3	1,306,3.5,1147868817																	
4	1,307,5.0,1147868828																	
5	1,665,5.0,1147878820																	
6	1,899,3.5,1147868510																	
7	1,1088,4.0,1147868495																	
8	1,1175,3.5,1147868826																	
9	1,1217,3.5,1147878326																	
10	1,1237,5.0,1147868839																	
11	1,1250,4.0,1147868414																	
12	1,1260,3.5,1147877857																	
13	1,1653,4.0,1147868097																	
14	1,2011,2.5,1147868079																	
15	1,2012,2.5,1147868068																	
16	1,2068,2.5,1147869044																	
17	1,2161,3.5,1147868609																	
18	1,2351,4.5,1147877957																	
19	1,2573,4.0,1147878923																	
20	1,2632,5.0,1147878248																	
21	1,2692,5.0,1147869100																	
22	1,2843,4.5,1147868891																	
23	1,3448,4.0,1147868480																	
24	1,3569,5.0,1147879603																	
25	1,3949,5.0,1147868878																	
26	1,4144,5.0,1147868898																	
27	1,4308,3.0,1147868534																	
28	1,4325,5.0,1147878122																	
29	1,4422,3.0,1147869048																	
30	1,4703,4.0,1147869223																	
31	1,4973,4.5,1147869080																	
32	1,5147,4.0,1147877654																	
33	1,5269,0.5,1147879571																	

And here is a glimpse of our movies.csv dataset:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	movieId,title,genres																	
2	1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy																	
3	2,Jumanji (1995),Adventure|Children|Fantasy																	
4	3,Grumpier Old Men (1995),Comedy|Romance																	
5	4,Waiting to Exhale (1995),Comedy|Drama|Romance																	
6	5,Father of the Bride Part II (1995),Comedy																	
7	6,Heat (1995),Action|Crime|Thriller																	
8	7,Sabrina (1995),Comedy|Romance																	
9	8,Tom and Huck (1995),Adventure|Children																	
10	9,Sudden Death (1995),Action																	
11	10,GoldenEye (1995),Action|Adventure|Thriller																	
12	11,"American President, The (1995)",Comedy|Drama|Romance																	
13	12,Dracula: Dead and Loving It (1995),Comedy|Horror																	
14	13,Balto (1995),Adventure|Animation|Children																	
15	14,Nixon (1995),Drama																	
16	15,Cutthroat Island (1995),Action|Adventure|Romance																	
17	16,Casino (1995),Crime|Drama																	
18	17,Sense and Sensibility (1995),Drama|Romance																	
19	18,Four Rooms (1995),Comedy																	
20	19,Ace Ventura: When Nature Calls (1995),Comedy																	
21	20,Money Train (1995),Action|Comedy|Crime|Drama|Thriller																	
22	21,Get Shorty (1995),Comedy|Crime|Thriller																	
23	22,Copycat (1995),Crime|Drama|Horror|Mystery|Thriller																	
24	23,Assassins (1995),Action|Crime|Thriller																	
25	24,Powder (1995),Drama|Sci-Fi																	
26	25,Leaving Las Vegas (1995),Drama|Romance																	
27	26,Othello (1995),Drama																	
28	27,Now and Then (1995),Children|Drama																	
29	28,Persuasion (1995),Drama|Romance																	
30	29,"City of Lost Children, The (Cité des enfants perdus, La) (1995)",Adventure|Drama|Fantasy|Mystery|Sci-Fi																	
31	30,Shanghai Triad (Yao a yao yao dao waipo qiao) (1995),Crime|Drama																	
32	31,Dangerous Minds (1995),Drama																	
33	32,Twelve Monkeys (a.k.a. 12 Monkeys) (1995),Mystery|Sci-Fi|Thriller																	
34	33,Wings of Courage (1995),Adventure|Romance|IMAX																	

A citation of where the dataset can be found,follows:
`https://grouplens.org/datasets/movielens/25m/`

4 Methods

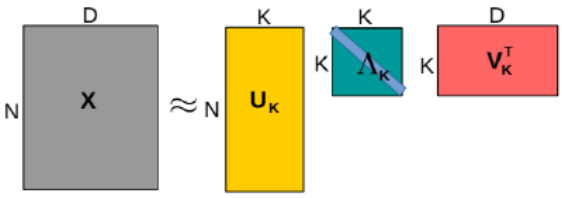
4.1 SVD

An algorithm that is suitable to make predictions through the use of arrays is the SVD algorithm.It is commonly used in recommendation systems to identify the latent structure in the user-item interaction matrix.The user-item interaction matrix is a matrix of user ratings for items, where each row represents a user and each column represents an item.SVD is a type of Matrix Factorization.Matrix factorization is the factorization of the user-item matrix into two lower-dimensional matrices whose product approximates the original matrix.The factorization process learns latent features that describe user and item interactions. These features can capture underlying aspects of the data,which can't be understood directly.The latent factors also show user tendencies,such as genre preferences in our case.Lastly,we recommend items to a user based on the items with the highest predicted interaction values that the user has not yet interacted with.Since the original user-item matrix A is usually sparse (not all users have rated all items), the SVD is often computed on a modified version of A where missing values have been filled in, or a regularized version of SVD is used (like SVD++), which can handle missing data.Our input for this algorithm was the ratings matrix that we calculated above.
In mathematical notation the SVD of a matrix A is represented as:

$$A = U\Sigma V^T$$

where U is an orthogonal matrix containing the left singular vectors of A
Σ is a diagonal matrix containing the singular values of A
V is an orthogonal matrix containing the right singular vectors of A
T denotes the transpose of a matrix

A visual representation of the algorithm breaking in 3 matrices,follows:



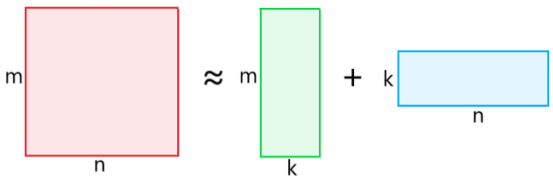
4.2 NMF

Non-negative Matrix Factorization or NMF is a method used to factorize a non-negative matrix, X, into the product of two lower rank matrices, A and B, such that AB approximates an optimal solution of X.This is an unsupervised learning algorithm used to reduce the dimensionality of data into lower-dimensional spaces.The input of NMF is the data matrix V and the output of Non-negative Matrix Factorization (NMF) typically consists of two matrices,the user matrix(W) and the item matrix(H).The product of these two matrices approximates the original matrix V.
Delving deeper,the W matrix represents the user profile or the latent features of the original data.Each row corresponds to a user (or data point), and each column corresponds to a latent feature.The H matrix contains the item profile or the coefficients for the latent features.Each row corresponds to a latent feature, and each column corresponds to an item.
The dimensions of W are m×n , where m is the number of original data points (e.g., users) and k is the number of latent features.The dimensions of H are k×n, where k is the number of latent features, and n is the number of items.
Our input for this algorithm was the ratings matrix that we calculated above.
In mathematical notation,the original matrix V is approximated as follows:

$$V \approx W \cdot H$$

where V is the original data matrix with dimensions m×n
W is the user matrix with dimensions m×k
H is the item matrix with dimensions k×n

A visual representation of the NMF Algorithm:



4.3 SGD

Application of SGD on the matrix factorization problem

Stochastic Gradient Descent (SGD) is an optimization method used to minimize an objective function by iteratively moving towards the minimum in small steps. In the context of matrix factorization, SGD is used to find two matrices P and Q that, when multiplied, approximate the original matrix R.
Our input for this algorithm was the ratings matrix that we calculated above.
1)Objective Function:

The objective function we want to minimize is the sum of squared errors over all user-item pairs, with an added regularization term to prevent overfitting:

$$L = \sum_{(u,i)\in K} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(\sum_u \|p_u\|^2 + \sum_i \|q_i\|^2 \right)$$

Here:

- r_{ui} is the actual rating of user u for item i,
- $\hat{r}_{ui}$ is the predicted rating, which is the dot product of p_u and q_i,
- λ is the regularization parameter,
- K is the set of all user-item pairs for which r_{ui} is known.

2) Gradient Calculation:
To minimize the loss function L using SGD, we compute the gradients with respect to the factors in P and Q

$$\frac{\partial L}{\partial p_{uk}} = -2(r_{ui} - \hat{r}_{ui})q_{ki} + 2\lambda p_{uk}$$

$$\frac{\partial L}{\partial q_{ki}} = -2(r_{ui} - \hat{r}_{ui})p_{uk} + 2\lambda q_{ki}$$

3)Update Rules:
We then update the factors in the opposite direction of the gradients.Where gamma is the learning rate, a positive scalar that determines the step size.

$$p_{uk} \leftarrow p_{uk} - \alpha \frac{\partial L}{\partial p_{uk}}$$

$$q_{ki} \leftarrow q_{ki} - \alpha \frac{\partial L}{\partial q_{ki}}$$

4) Iterative Process:
These updates are applied iteratively, typically selecting a random user-item pair at each step (hence the ”stochastic” in SGD):

$$p_{uk} \leftarrow p_{uk} + \gamma \left(2(r_{ui} - \hat{r}_{ui})q_{ki} - \lambda p_{uk}\right)$$

$$q_{ki} \leftarrow q_{ki} + \gamma \left(2(r_{ui} - \hat{r}_{ui})p_{uk} - \lambda q_{ki}\right)$$

This process continues until the loss function converges to a minimum or a maximum number of iterations is reached. The result are matrices P and Q that provide a low-rank approximation of R capturing the main structure in the data while reducing noise and overfitting.

4.4 GPT-like Recommendations

The GPT-like approach for generating recommendations in this context is a simplified simulation meant to mimic how a language model, such as GPT, might process text-based user queries to make item recommendations. This approach does not involve an actual GPT model or natural language processing; instead, it’s a rule-based simulation using Python and the user-item ratings data. Here’s a detailed breakdown:

1. Data Preparation:
User-Item Ratings Matrix: We start with a synthetic dataset representing user-item interactions. This dataset is a matrix where each entry indicates a user’s rating for an item on a scale of 0 to 5.
Textual Representation of User Preferences: Each user’s preferences are converted into a textual format. For example, if ’User1’ rated ’Item1’, ’Item4’, and ’Item6’ highly, their preferences are summarized as ”User1 likes Item1, Item4, Item6.”
2. Simulating User Queries
Query Formation: The approach involves simulating a user query in a natural language format, like ”I liked Item1,

Item4, and Item6. What other items would you recommend?
Extraction of Liked Items: The code extracts the items mentioned in the query (e.g., Item1, Item4, Item6) to understand the user's expressed preferences
3. Generating Recommendations
Identifying Similar Users: The algorithm identifies other users who have also liked the same items. This step is based on the assumption that users with similar tastes will likely appreciate the same items.
Aggregate Preferences of Similar Users: It then aggregates the ratings of these similar users to identify other items they liked.
Recommendation of Items: The items most favored by these similar users, which the querying user hasn't rated yet, are recommended as potential items of interest.
4. Limitations and Notes
Simplicity: This method is quite rudimentary compared to actual GPT capabilities. A real GPT model, if applied to this task, would involve more sophisticated natural language understanding and generation, potentially considering nuanced user preferences and contextual information.
No Actual Language Model: The process doesn't involve any machine learning or language model predictions. It's a rule-based approach imitating how a recommendation might be made based on textual data.
Focus on Explicit Preferences: This approach primarily considers explicit preferences mentioned by the user, and it might not capture subtle or latent preferences.

Our input for this algorithm was the user,for whom we want to recommend movies,the ratings dataframe and the number of recommendations we wish to suggest.

The GPT-like method here is a basic stand-in for how a complex language model might approach recommendation generation in a text-based, conversational context. In practice, GPT or similar models would likely offer more nuanced and context-aware recommendations by understanding and generating human-like text.

4.5 k-NN Recommendations

To apply a K-Nearest Neighbors (KNN) algorithm for collaborative filtering using the synthetic dataset, we'll focus on user-based collaborative filtering. In this approach, we find the 'K' nearest neighbors (i.e., users with similar preferences) for a target user and use their ratings to predict the target user's preferences.
Here's the general process:
1. Preprocessing: Normalize the ratings matrix so that each user's rating is centered around 0. This helps in reducing the impact of varying rating scales between different users.
2. Similarity Computation: Calculate the similarity between users. Commonly used metrics are cosine similarity or Pearson correlation.
3. Finding Nearest Neighbors: For each user, find the 'K' users most similar to them.
4. Predicting Ratings: For a target user, predict the ratings for items they haven't rated yet based on the ratings of the nearest neighbors.
5. Generating Recommendations: Recommend items with the highest predicted ratings to the user.
Specifically,our code performs the following steps:
1. Data Preparation: Creates a synthetic user-item ratings matrix.
2. Normalization: Centers each user's ratings around 0 to normalize the data.
3. Cosine Similarity Calculation: Computes the cosine similarity between users to find similarities.
4. KNN Model: Applies a KNN algorithm to find the 'K' nearest neighbors for each user.
5. Recommendation Generation: Predicts ratings for unrated items based on neighbors' preferences and recommends items with the highest predicted ratings.
The final output provides personalized item recommendations for a specific user based on the preferences of their nearest neighbors.
Our input for this algorithm was the first 150 users,along with the ratings of every movie those users have seen,the user for whom we want to recommend movies,the cosine-similarity matrix,and the number of neighbors we want to examine.
In mathematical notation,the predicted rating from a user u to an item i ($\hat{R}_{ui}$) can be computed, for example, as the weighted average of the ratings given by the nearest neighbors:

$$\hat{R}_{ui} = \frac{\sum_{v \in N_u} |S(u,v)| \cdot R_{vi}}{\sum_{v \in N_u} S(u,v)}$$

where S(u,v) is the similarity between 2 users u and v

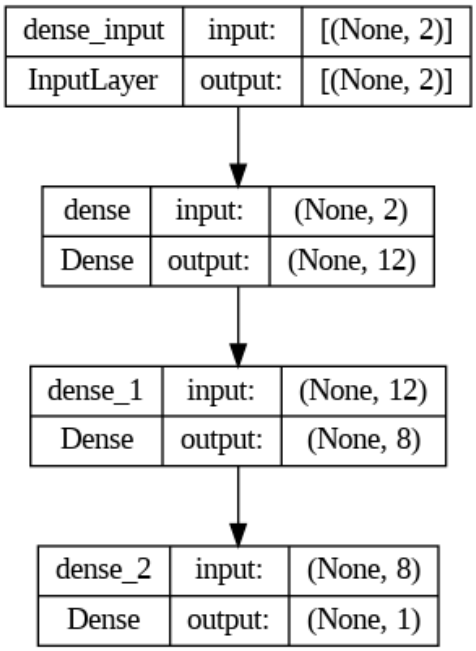
N_u represents the k nearest neighbors of user u based on similarity
R_{vi} represents the rating given by user v to item i

4.6 Keras-Implementation

One of our implementations for creating a recommender system was a Neural Network(NN) through Keras.Our NN is comprised of 3 layers.The first layer is of type ”Dense” with a ReLU activation function.Dense implements the operation:output = activation(dot(input, kernel) + bias)where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if useBias is True).The input of the first layer is the UserID and the MovieID and it has 12 outputs,which are connected with the inputs of the second layer.The second layer has 8 outputs which are connected with the third layer.Lastly,the output of the third layer is a rating.After we get that rating,we connect it with the corresponding MovieId to complete our recommendation.We defined MSE(Mean Squared Error) as the metric for our loss function.In order to reduce the error of the loss function,we used the ADAM optimizer.Therefore,through the back-propagation,the neural network gets trained and the weights reach their optimal value.

Essentially,after the neural network’s training is complete,the movies with the highest predicted ratings are recommended to the user.The training of the neural network happened with every user.The weights are updated in such way,so that the combination of every user with every item is used for training.

Visually, our Neural Network looks like this:



4.7 The Alphas Method

The Alphas Method is a recommendation method that we came up with.The main question that we had to answer,and the question in which we based our method on,is the following:
1)Which movie genre does a certain user like?
The second question is the following:
We have a user who rated 10 movies of the same genre with 1 star and 2 movies from another genre with 5 stars. Which of the two genres does the user like most?The answer is vague and that brings us to the following question:Is the frequency or the rating of a certain genre,more important in determining what the user likes?
The procedure that we followed to answer the above question is the following:
1)We collected the movies that a certain user has seen
2)We then created the following dataframe:

movieId	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War
296	0.0	0.0	0.0	0.0	5.0	5.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0
6370	0.0	0.0	0.0	0.0	4.5	0.0	0.0	4.5	0.0	0.0	0.0	0.0	0.0	4.5	0.0	0.0	0.0
6377	0.0	4.0	4.0	4.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6539	3.5	3.5	0.0	0.0	3.5	0.0	0.0	0.0	3.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6711	0.0	0.0	0.0	0.0	5.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0

In the above dataframe,the rows are the movies(MovieId) that a certain user has seen,the columns of the dataframe are the movie genres and the elements of the dataframe are the ratings that the user gave to a movie.If a movie belongs in multiple genres,the rating of the movie is put in all of the genres that the movie belongs.

3)The formula,with which we calculate the liking of a certain user to a certain genre,is the following:
We sum the reviews of a certain genre and we divide that number with the sum of all reviews in the dataframe.We do that for every genre.What we get is the following array:

	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi
0	0.0275	0.068333	0.013333	0.019167	0.148333	0.054167	0.003333	0.341667	0.03	0.005833	0.005833	0.030833	0.023333	0.125	0.030833

The biggest value in that array signifies the category that the user likes the most.That is the category,from which we will recommend movies.Then,we apply a certain threshold,so that we cut out the movies that have only a few votes.From the movies that are left,we recommend the movies that have the highest mean value.

5 Experiments/Results/Discussion

5.1 SVD

Our hyperparameter in this algorithm was the number of latent factors(k).After trial and error,we found out that the k value that suits us was when k = 15.It is worth noting that we are using SVDS(SVD for sparse arrays) because our array is sparse.Therefore,we obtain better results in our case.
Here is a screenshot of our output:

	userId	movieId	rating
1843	1	4973	1.084015
163	1	296	1.045422
2079	1	6711	0.993383
1722	1	4226	0.944523
1991	1	6016	0.915871
1287	1	2858	0.891056
288	1	527	0.862655
2167	1	7361	0.838025
324	1	593	0.830941
1328	1	2959	0.812344

And here are the errors that came up from our metrics:
RMSE: 0.43735932964969926
MSE: 0.1912831832316343
MAE: 0.1523274998334627

5.2 NMF

Our hyperparameter in this algorithm was the number of n_components(n).After trial and error,we found out that the n value that suits us was when n = 10.Something to note is that random initialization can sometimes lead to faster convergence, but it might also result in different solutions on different runs.We decided to take the small risk of different solutions,and to keep the feature of random initialization.
Here is a screenshot of our output:

	userId	movieId	rating
1287	1	2858	0.880042
163	1	296	0.864523
1328	1	2959	0.845846
1722	1	4226	0.809435
1991	1	6016	0.803009
324	1	593	0.754779
2713	1	58559	0.750620
1063	1	2329	0.742745
1843	1	4973	0.737129
2735	1	60069	0.718238

And here are the errors that came up from our metrics:
RMSE: 0.502634327080889
MSE: 0.25264126676005805
MAE: 0.14983825285608293

5.3 SGD

Here is a screenshot of our output:

	userId	movieId	rating
3195	1	135532	5.097703
2738	1	60103	5.092287
853	1	1892	5.009409
2152	1	7234	4.972377
36	1	50	4.899531
288	1	527	4.873461
1704	1	4144	4.866823
354	1	668	4.832679
251	1	457	4.788210
432	1	912	4.786171

And here are the errors that came up from our metrics:
RMSE: 0.8655834144007515
MSE: 0.749234647285663
MAE: 2.7063016811786507

Our hyperparameter in this algorithm was the learning rate(lr).After trial and error,we found out that the n value that suits us was when lr = 0.0002.Also,we found out that beta = 0.02 and steps = 100.Learning rate is an important parameter because it shows how ”sharply” the model converges and also it controls how the values of the matrices P and Q change as the algorithm runs.

5.4 GPT-like Recommendations

Here is a screenshot of our output:

	movieId	rating
1070	27728	5.0
1105	33270	5.0
1361	98491	5.0
1363	98607	5.0
203	923	5.0
1368	103228	5.0
1059	26903	5.0
1060	27156	5.0
505	2395	5.0
1211	52885	5.0

And here are the errors that came up from our metrics:
MSE GPT: 0.7433619281045752

RMSE GPT: 0.8621843933315977
MAE GPT: 0.6936274509803921

5.5 k-NN Recommendations

Here is a screenshot of our output:

	movieId	rating
1779	2959	1.566358
1402	2329	1.515796
2405	4226	1.501142
1569	2571	1.475541
3128	6874	1.437188

And here are the errors that came up from our metrics:
KNN MSE: 11.983475789750738
KNN RMSE: 3.4617157291942298
KNN MAE: 3.3518419799301196

We notice that the error in k-NN algorithm is much bigger compared to the previous algorithms.This happens probably because our ratings matrix is sparse.In collaborative filtering, a ratings matrix is often sparse because not all users rate all items. The sparsity of the matrix can lead to challenges in finding similar users or items, as there may be limited overlapping ratings.A second reason this might happening,is that the sparse array creates the cold start problem.In the cold start problem,the system struggles to make accurate recommendations for new users or items with few or no ratings.So,the k-NN algorithm does not work well on our dataset.

5.6 Keras-Implementation

Here is a screenshot of our output:

	userId	movieId	rating
17239881	1	1228	5.003172
17244522	1	1292	5.002780
17266875	1	1701	5.001251
4862066	1	1676	5.001193
17280695	1	1834	5.001052
17277020	1	1810	5.001001
17275877	1	1794	5.000991
17261693	1	1569	5.000940
17272779	1	1719	5.000598
17275192	1	1726	5.000559

And here are the errors that came up from our metrics:
MSE Keras: 1.6664779565189747e-06
RMSE Keras: 0.001290921359540919
MAE Keras: 0.001012176275253296

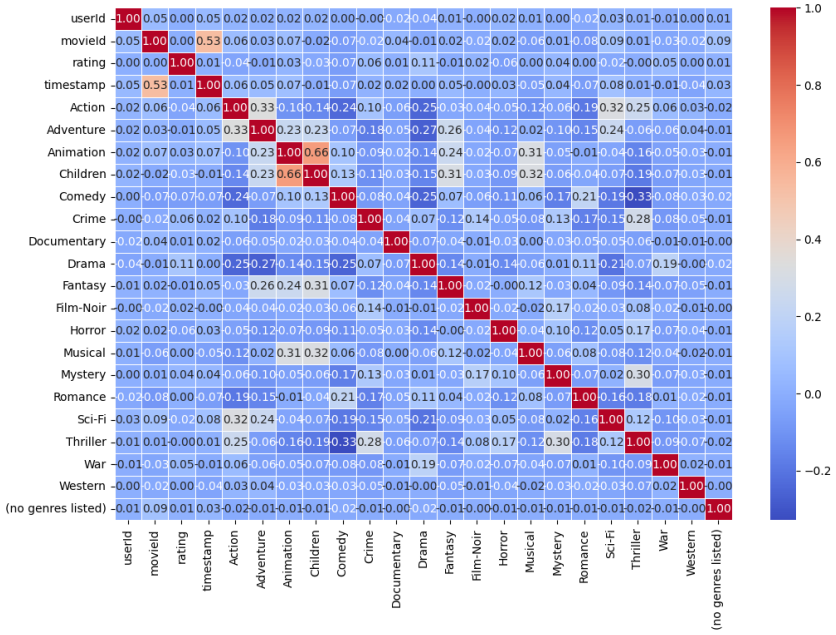
5.7 The Alphas Method

Here is a screenshot of our output:

	userId	movieId	title	genres	rating
480515	318	2019	Shawshank Redemption, The (1994)	Crime|Drama	4.413570
0	296	1	Pulp Fiction (1994)	Comedy|Crime|Drama|Thriller	4.189512
204319	6210	1	City of God (Cidade de Deus) (2002)	Action|Adventure|Crime|Drama|Thriller	4.181887
110351	1217	1	Ran (1985)	Drama|War	4.142053
192301	2761	1	Nights of Cabiria (Notti di Cabiria, Le) (1957)	Drama	4.137364
134154	1290	1	Bridge on the River Kwai, The (1957)	Adventure|Drama|War	4.099160
119671	1237	1	Seventh Seal, The (Jungfrukällan, Det) (1957)	Drama	4.069324
276970	1327	1	Persona (1966)	Drama	4.063058
217329	6147	1	Wild Strawberries (Smultronstället) (1957)	Drama	4.060701
79972	899	1	Three Colors: Red (Trois couleurs: Rouge) (1994)	Drama	4.072967

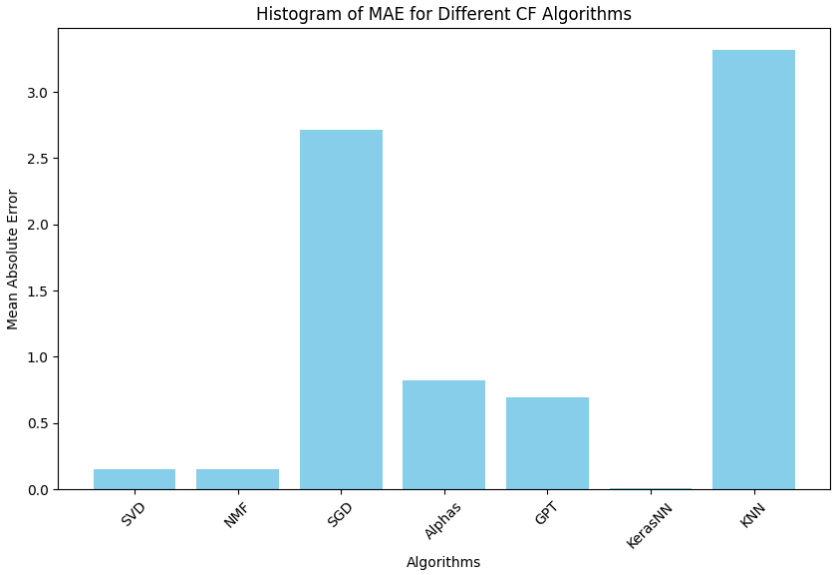
And here are the errors that came up from our metrics:
RMSE Alphas: 0.9850522236482879
MSE Alphas: 0.9703278833144368
MAE Alphas: 0.8210276744297015

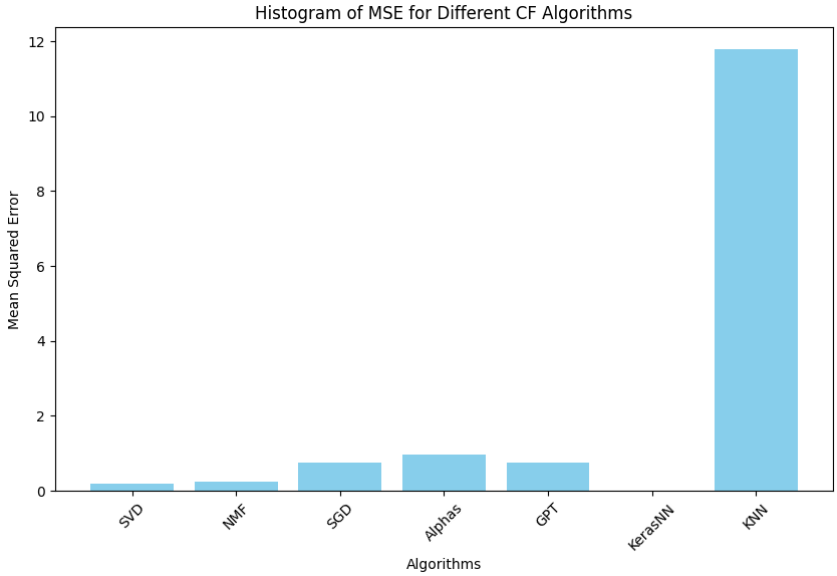
5.8 Heatmap

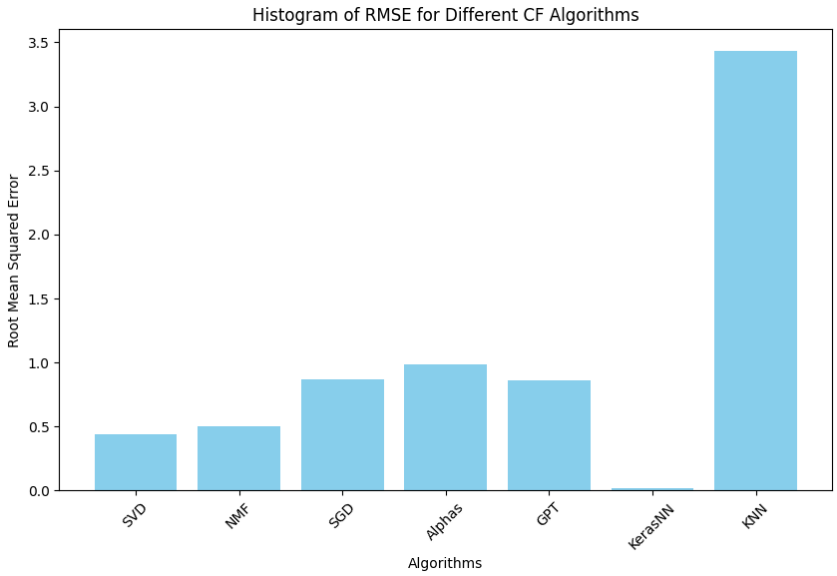


You can also find the heatmap in higher definition in our code file.

5.9 Diagrams of Performance







6 Conclusion

To sum up,our project was about recommendation systems and different ways with which we can build a system like this.It was a really interesting project and our knowledge regarding Recommendation Systems and ML in general,has increased.We studied 7 methods and after we implemented them,we measured the performance for each and every one system,compared them and comprehended them.
As our future work,we can develop the method that we came up with (the alphas) even more.We can also try to reduce our errors even further by applying feature engineering and other techniques.

7 Contributions

All members worked equally in all parts of the project.

8 References/Bibliography/Libraries

8.1 References/Bibliography

1. https://www.researchgate.net/profile/Mahendra-Prasad-3/publication/314132367_Machine_Learning_Algorithms_for_Recommender_System_-_a_comparative_analysis/

links/58c8047daca2723ab167256e/Machine-Learning-Algorithms-for-Recommender-System-a-com
pdf

2.https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6993356

3.https://arxiv.org/ftp/arxiv/papers/2106/2106.10679.pdf

4.https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4052704

5.https://www.vikas.sindhwani.org/recommender.pdf

6.Recommendation Systems using Different Methods https://piazza.com/class/llug8fzp1x63qa/
post/84

8.2 Libraries

```
from google.colab import drive
from scipy.stats import zscore
fromscipy.stats.mstats import winsorize
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean _absolute _error
from sklearn.metrics import mean _squared _error
from scipy.sparse.linalg import svds
from sklearn.metrics import mean_squared_error
from sklearn.decomposition import NMF
from numpy import loadtxt
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import plot_model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics.pairwise import cosine_similarity
from pydoc import help
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
from sklearn.discriminant _analysis import LinearDiscriminantAnalysis
from scipy import stats
from IPython.display import display, HTML
import math
from google.colab import drive
from scipy.stats import zscore
from scipy.stats.mstats import winsorize
```