Capstone Project –Frank Li, Yuyan Yang, Yuzhuo Huang

- Link to our group's GitHub repository:

https://github.com/nyu-big-data/capstone_94/tree/main

- List of top 100 most similar pairs (include a suitable estimate of their similarity for each pair), sorted by similarity

To identify and evaluate the top 100 most similar pairs of users, we utilized a combination of data loading, transformation, and similarity measurement techniques within a Spark environment. First, we loaded user ratings, tags, and movie metadata from HDFS and merged these datasets to create a comprehensive rating history. This data was then transformed into a user-movie matrix, which served as the input for our similarity algorithm. We employed a MinHash-based locality-sensitive hashing (LSH) approach, which efficiently approximates the Jaccard similarity between users. By tokenizing movie titles and transforming them into feature vectors, we fitted the MinHashLSH model and transformed the dataset to obtain binary hash buckets. Using this transformed data, we performed an approximate similarity join to find pairs of users with similar movie-watching habits. We then filtered the results to remove duplicate and self-pairs, sorting the remaining pairs based on their Jaccard distance. Finally, we selected the top 100 pairs with the highest similarity scores, providing an evaluation of user similarity within the dataset.

rank	User ID#	User ID #	Jaccard distance
1	479	571	0.5833333333333333
2	571	151	0.5833333333333333
3	136	44	0.5833333333333333
4	34	597	0.5833333333333333
5	339	34	0.5833333333333333
6	44	353	0.5833333333333333
7	151	19	0.5833333333333333
8	325	20	0.5714285714285714
9	217	496	0.5714285714285714

10	20	607	0.5714285714285714
11	73	2	0.5714285714285714
12	21	61	0.5714285714285714
13	474	2	0.5714285714285714
14	361	21	0.5714285714285714
15	553	9	0.5714285714285714
16	406	23	0.5714285714285714
17	476	14	0.5714285714285714
18	435	23	0.5714285714285714
19	19	508	0.5714285714285714
20	432	2	0.5714285714285714
21	177	7	0.5714285714285714
22	11	505	0.5714285714285714
23	32	486	0.5714285714285714
24	518	33	0.5714285714285714
25	254	34	0.5714285714285714
26	34	279	0.5714285714285714
27	238	37	0.5714285714285714
28	62	39	0.5714285714285714
29	39	238	0.5714285714285714
30	42	484	0.5714285714285714
31	410	47	0.5714285714285714
32	419	51	0.5714285714285714
33	51	436	0.5714285714285714
34	54	238	0.5714285714285714

35	96	61	0.5714285714285714
36	191	61	0.5714285714285714
37	487	61	0.5714285714285714
38	409	62	0.5714285714285714
39	595	68	0.5714285714285714
40	547	72	0.5714285714285714
41	73	338	0.5714285714285714
42	74	527	0.5714285714285714
43	479	75	0.5714285714285714
44	76	382	0.5714285714285714
45	425	79	0.5714285714285714
46	80	221	0.5714285714285714
47	80	255	0.5714285714285714
48	82	293	0.5714285714285714
49	82	578	0.5714285714285714
50	85	353	0.5714285714285714
51	85	468	0.5714285714285714
52	602	87	0.5714285714285714
53	90	606	0.5714285714285714
54	91	240	0.5714285714285714
55	457	91	0.5714285714285714
56	265	92	0.5714285714285714
57	92	599	0.5714285714285714
58	115	93	0.5714285714285714
59	435	96	0.5714285714285714

60	127	102	0.5714285714285714
61	475	103	0.5714285714285714
62	172	105	0.5714285714285714
63	136	108	0.5714285714285714
64	108	345	0.5714285714285714
65	484	114	0.5714285714285714
66	115	546	0.5714285714285714
67	115	547	0.5714285714285714
68	117	380	0.5714285714285714
69	124	449	0.5714285714285714
70	323	127	0.5714285714285714
71	127	429	0.5714285714285714
72	384	136	0.5714285714285714
73	534	138	0.5714285714285714
74	368	139	0.5714285714285714
75	373	143	0.5714285714285714
76	148	260	0.5714285714285714
77	536	151	0.5714285714285714
78	159	509	0.5714285714285714
79	164	578	0.5714285714285714
80	238	165	0.5714285714285714
81	169	192	0.5714285714285714
82	232	169	0.5714285714285714
83	261	172	0.5714285714285714
84	484	172	0.5714285714285714

85	172	599	0.5714285714285714
86	179	287	0.5714285714285714
87	435	182	0.5714285714285714
88	380	190	0.5714285714285714
89	496	194	0.5714285714285714
90	195	466	0.5714285714285714
91	413	200	0.5714285714285714
92	578	202	0.5714285714285714
93	238	208	0.5714285714285714
94	323	210	0.5714285714285714
95	217	281	0.5714285714285714
96	232	419	0.5714285714285714
97	289	234	0.5714285714285714
98	302	238	0.5714285714285714
99	392	238	0.5714285714285714
100	538	238	0.5714285714285714

- A comparison between the average pairwise correlations between these highly similar pair and randomly picked pairs

- Average Jaccard Distance for Top 100: 0.5717857142857138
- Average Jaccard Distance for Random 100: 0.09769841269841269

- Documentation of how your train/validation splits were generated

The dataset was first split into training and testing sets using a 70:30 ratio. This initial split ensured that the majority of the data was used for training, while a substantial portion was reserved for testing the model's performance on unseen data. We used a fixed random seed (seed=0) to guarantee reproducibility.

The training set obtained from the initial split was further divided into training and validation sets using an 80:20 ratio. This step allowed us to create a validation set from the training data, which was used for tuning model parameters and preventing overfitting.

- Any additional pre-processing of the data that you decide to implement

Initially, we loaded the user ratings, tags, and movie metadata from HDFS and merged these datasets to form a comprehensive rating history. We then transformed this data into a user-movie matrix using a pivot table, which allowed us to efficiently handle missing values by filling them with zeros. This transformation facilitated the subsequent tokenization of movie titles, which was essential for converting textual data into a format suitable for hashing. By applying a Tokenizer and HashingTF, we transformed the movie titles into feature vectors, preparing the data for similarity measurement. These pre-processing steps ensured that the dataset was clean, structured, and ready for the MinHashLSH model to identify similar user pairs effectively.

- Evaluation of popularity baseline

To establish a baseline for our model's performance, we evaluated a popularity-based recommendation system. In this approach, movies were recommended to users based on their overall popularity, determined by the number of ratings and average rating scores.

By comparing the results of the popularity baseline with our collaborative filtering model, we aimed to demonstrate the added value of using more sophisticated techniques for personalized recommendations. We first used MAP(Mean Average Precision) to evaluate the model, getting MAP = 1.8035459416992854e-05, a pretty small number. This indicates that, on average, the recommendations made by the popularity model are not very effective at ranking relevant items highly for users. The low MAP suggests that while some relevant items might be recommended, they are often not ranked high in the list, since the popularity model does not align well with individual user preferences. Since MAP takes a long time to run across all users, and the cluster kills many times, we used precision and recall in both popularity based model and latent factor model as the evaluation method to compare results. We got Average Precision@10 of 0.0006611570247933885, and Average Recall@10 of 6.155355721137302e-05.

The following are top 20 Movies sorted and recommended by popularity based model according to each movie's average rating. Movies are ranked in alphabetical order. The number of movies with a rating of 5 is not negligible, so we decided to evaluate based on the top 20 moves based on ascending order for ratings and alphabetical order.

Title	Avg Rating
'Salem's Lot (2004)	5.0

7th Voyage of Sinbad, The (1958)	5.0
Adam's Rib (1949)	5.0
Before Night Falls (2000)	5.0
Best of Youth, The (La meglio gioventù) (2003)	5.0
Bill Hicks: Revelations (1993)	5.0
Cosmic Scrat-tastrophe (2015)	5.0
Dead Man's Shoes (2004)	5.0
George Carlin: You Are All Diseased (1999)	5.0
Goodbye Charlie (1964)	5.0
Gulliver's Travels (1939)	5.0
Heidi Fleiss: Hollywood Madam (1995)	5.0
Human Condition III, The (Ningen no jôken III) (1961)	5.0
Lady Jane (1986)	5.0
Martin Lawrence Live: Runteldat (2002)	5.0
Paper Birds (Pájaros de papel) (2010)	5.0
Radio Day (2008)	5.0
Rain (2001)	5.0
Reform School Girls (1986)	5.0
Scooby-Doo! Abracadabra-Doo (2010)	5.0

- Documentation of latent factor model's hyper-parameters and validation

The hyper-parameters included the rank of the factorization (number of latent factors) and the regularization parameter, which controls the complexity of the model to prevent overfitting. We tested the model with different combinations of rank (5, 10, 15, and 20) and regularization parameters (0.01, 0.1, 1.0, and 10.0). The performance of each configuration was evaluated using the validation Root-Mean-Square Error (RMSE), with the goal of minimizing this metric. The best performance was achieved with a rank of 20 and a regularization parameter of 0.1, resulting in a validation RMSE of 0.9206251341827478.

- Evaluation of latent factor model

Validation Root-Mean-Square Error (RMSE)

The validation RMSE for different hyper-parameter settings were as follows:

- Rank: 5
    - Regularization: 0.01, Validation RMSE: 1.0456472257478628
    - Regularization: 0.1, Validation RMSE: 0.9210528817245397
    - Regularization: 1.0, Validation RMSE: 1.3343259924465753
    - Regularization: 10.0, Validation RMSE: 3.6673397418096076
- Rank: 10
    - Regularization: 0.01, Validation RMSE: 1.1034618719763254
    - Regularization: 0.1, Validation RMSE: 0.9223407943835541
    - Regularization: 1.0, Validation RMSE: 1.3343273491276393
    - Regularization: 10.0, Validation RMSE: 3.6673397418096076
- Rank: 15
    - Regularization: 0.01, Validation RMSE: 1.1308962824615387
    - Regularization: 0.1, Validation RMSE: 0.9217413060696134
    - Regularization: 1.0, Validation RMSE: 1.3343278243720127
    - Regularization: 10.0, Validation RMSE: 3.6673397418096076
- Rank: 20
    - Regularization: 0.01, Validation RMSE: 1.1479256198181194
    - Regularization: 0.1, Validation RMSE: 0.9206251341827478
    - Regularization: 1.0, Validation RMSE: 1.3343260035890714
    - Regularization: 10.0, Validation RMSE: 3.6673397418096076

The optimal configuration was found to be a rank of 20 and a regularization parameter of 0.1, achieving the best validation RMSE of 0.9206251341827478.

Test Root-Mean-Square Error (RMSE) and Performance Metrics

Upon evaluating the latent factor model on the test set, we achieved a test RMSE of 0.9138640787358804. Additional performance metrics included a Precision@5 of 0.29055724274313155 and a Recall@5 of 0.29055724274313155, indicating that the model effectively captures user preferences and provides relevant recommendations. Also, compared with the baseline model, the popularity based model, the precision and recall have a huge improvement, suggesting the latent factor model's superior performance in making relevant recommendations.

- Distribution of works
    - Frank worked on data pre-processing, train/validation splits, computed similarity scores, implemented the latent factor model, performed hyper-parameter tuning, and managed the GitHub repository.
    - Yuyan wrote the section on pre-processing and data splitting, train/validation splits, hyper-parameter tuning, and similarity pairs and their comparison.
    - Kina conducted the evaluation of the popularity baseline model, and wrote the section on evaluating and comparing latent factor model with the baseline model.