International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 4, ISSN No. 2455-2143, Pages 223-226
Published Online August 2020 in IJEAST (http://www.ijeast.com)

DESIGN AND ANALYSIS OF COLLABORATIVE FILTERING BASED RECOMMENDATION SYSTEM

Sonali Suryawanshi
Department of IT
Walchand College of Engineering, Sangli,
Maharashtra, India

Manish Narnaware
Department of IT
Walchand college of Engineering, Sangli
Maharashtra, India

***Abstract*— Recommender Engine is a specific type of smart system that uses old user feedback on products and/or additional information to make useful product recommendations. This assumes a key job in a wide scope of utilization, including web-based shopping, e-business administrations, and social ecommerce networking. Collaborative sifting (CF) is the most well-known methodologies utilized for suggestion frameworks; however, CF experiences full cold start (CCS) issue where no appraising record is accessible and with Incomplete Cold Start (ICS) issues where there are just few rating records accessible for some new things or application clients. Therefore, the recommendation algorithms for collaborative filtering are useful and play a vital role in businesses to reach out to new users and promote their services and products. This paper introduces a new cooperative filtering recommendation algorithm based on dimensionality reduction called Singular Value Decomposition (SVD) used to cluster related users and reduce dimensionality. These method and concept are continuously being used and referred in order to attain an increased and enhanced accuracy over the present Netflix system. This paper is working with Netflix's prize dataset, we use the incremental SVD approach to predict movie ratings based on previous user preferences. Different experiments are conducted to see the effect of various parameters on the algorithm's performance.**

Keywords— **Recommendation System, Collaborative Filtering, dimension reduction and Singular Value Decomposition (SVD).**

I. INTRODUCTION

Recommended System is the last case of the standard awesome gigantic change in computing learning. applications, for example, web-based business, video, internet music, and gaming, and even online relationship practice a comparative strategy that has huge volumes of information to satisfy a client's needs in a customized manner. while many of the work in advice focuses on algorithms, another technique of an advice machine is that it has a massive impact. for instance, together with new statistics sources or representations (features) to a current algorithm. we will utilize netflix as a decent model for depicting the utilization of information, models, and distinctive personalization systems. Step by step instructions to gauge the accomplishment of a given personalization technique is some other basic thing that is thought about. Root Mean Squared Error (RMSE) was at one time the disconnected correlation metric picked for the Netflix Prize.

Collaborative Filtering is the technique for foreseeing a client's advantages dependent on all clients ' past inclinations dependent on the possibility that clients who have enjoyed comparative things in the past will in general support comparable things later on. Extensive work on this topic has been done in the literature, and Netflix is currently running a cooperative screening contest for movies. The goal of the competition is to increase performance by 10 percent over its current algorithm. Different teams in the competition have shown Singular Value Decomposition (SVD) to perform well for this goal, and nearly all groups currently in top positions in the leader board employ a form of SVD.

II. PROPOSED ALGORITHM

A. Singular Value Decomposition(SVD) algorithm –

Data sparsity and high dimensionality are recurring problems in RS. Therefore, dimensionality reduction is an urgent problem to be solved at present, and SVD namely a particular realization of the MF algorithms, is a powerful technique for dimensionality reduction an original rating matrix X_{mXn} can be decomposed into U, S and V according to SVD technology as follows

$$X_{mXn} = U_{mXr} S_{rXr} V^{T}_{rXn} \quad (1)$$

Each column of U is called a left singular vector, S is a diagonal matrix, and the diagonal values are arranged from large to small, which are called singular values; each row of V^T is called the right singular vector

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 4, ISSN No. 2455-2143, Pages 223-226
Published Online August 2020 in IJEAST (http://www.ijeast.com)

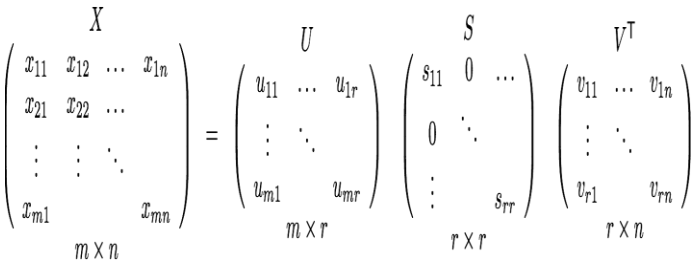


Fig. 1. SVD matrix

U and V matrix is orthogonal matrix, X is user's rating matrix, U is user's features matrix, and S is diagonal matrix of singular values, V^T is movies features matrix, U represents hoe much users "like" each feature, V^T represents how relevant each feature is to each movie.

The value of the diagonal on the matrix S is indeed the square root of RR^T or R^TR. For instance, a matrix decomposition process of SVD. The dimension of the initial matrix R is reduced, which is represented by using U, S, and V. Among them, U reflects the user information, V reflects the item information, and S reflects the importance of the feature. We select the first 4 features, which take up more than 95% of the original energy. Finally, $\hat{R}$ approximates to the real matrix R. In general, S is a r×r diagonal matrix, where k = min(m,n). R is approximated with $\hat{R}$ given by R ≈ $\hat{R}$ = U $\hat{\sum}$ V, and $\hat{\sum}$ is the k-rank approximation of ∑.

In RS, SVD is used for dimensionality reduction, and it can also be used directly for prediction tasks. The prediction process is as follows:

Step 1: Covert the rating matrix to the new dense matrix D. The user-item rating matrix $X_{mXn}$is mapped to the dense matrix D_{mXn}using SVD techniques i.e., for finding the new coordinates of users and items in the matrix D_{mXn} we convert raw data to the k-dimensions space as follows:

$$U_{Trans} = X_{mXn}\, V_{nXk} \sum^{-1}{}_{kXk} \qquad (2)$$

$$V_{Trans} = X_{mXn}\, U_{nXk} \sum^{-1}{}_{kXk} \qquad (3)$$

Where U_{Trans} and V_{Trans} are new coordinates of users and items in the k dimensions space. For instance, the matrix is denoted as R, which can be decomposed into U, V, and ∑. We can obtain the approximation of R by taking the first 2-dimensional data.

Step 2: Normalize the rating matrix D. The matrix D is normalized employing Z-score to the Z_{mXn} by $Z_{ij}^{(u)} = \frac{U_{ij} - \overline{U_i}}{\pi_i}$ and $Z_{ij}^{(i)} = \frac{I_{ij} - \overline{I_j}}{\sigma_j}$ .Here $\overline{U}$ and π denote the average ratings and standard deviation for users, respectively, and $\overline{I}$ and σ denote the average ratings and standard deviation for items, respectively.

Step 3: Apply SVD method on the matrix Z. i.e., the matrix Z is decomposed using SVD to obtain the new U, S and V.

Step 4: Obtain an approximation of Z. According to the low-rank matrix U, S and V, we can obtain a new matrix, denoted by$\hat{Z}$.

Step 5: Predict the unknown ratings. We can predict the unknown ratings based on $\hat{r}_{ij}^{(u)} = \overline{U_i} + \pi_i Z_{ij}^{(u)}$ or $\hat{r}_{ij}^{(i)} = \overline{I_j} + \sigma_j Z_{ij}^{(i)}$.

III. Experiment and Result

1. Problem Description

Netflix is all about connecting people to the movies they love. To help customers find those movies, they developed world-class movie recommendation system: CinematchSM. Its job is to predict whether someone will enjoy a movie based on how much they liked or disliked other movies. Netflix use those predictions to make personal movie recommendations based on each customer's unique tastes. Netflix provided a lot of anonymous rating data, and prediction accuracy and predict the rating that a user would give to a movie that he has not yet rated and also Minimize the difference between predicted and actual rating.

2. Converting / merging whole data to required format

This process is to convert whole data into required format i.e. we are checking for null values, removing duplicate values, checking of overall data like total number of ratings, total number of movies, and total number of users.

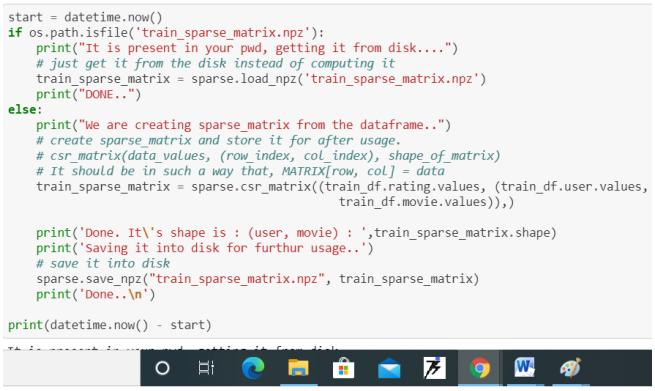
```
start = datetime.now()
if not os.path.isfile('data.csv'):
    # Create a file 'data.csv' before reading it
    # Read all the files in netflix and store them in one big file('data.csv')
    # We re reading from each of the four files and appendig each rating to a global file 'train.csv'
    data = open('data.csv', mode='w')

    row = list()
    files=['data_folder/combined_data_1.txt','data_folder/combined_data_2.txt',
           'data_folder/combined_data_3.txt', 'data_folder/combined_data_4.txt']
    for file in files:
        print("Reading ratings from {}...".format(file))
        with open(file) as f:
            for line in f:
                del row[:] # you don't have to do this.
                line = line.strip()
                if line.endswith(':'):
                    # All below are ratings for this movie, until another movie appears.
                    movie_id = line.replace(':', '')
                else:
                    row = [x for x in line.split(',')]
                    row.insert(0, movie_id)
                    data.write(','.join(row))
                    data.write('\n')
        print("Done.\n")
    data.close()
print('Time taken :', datetime.now() - start)
```

Fig. 2: preprocessing of data

3. Create the sparse matrix from dataframe

Here we are creating the sparse matrix and checking the sparsity of matrix has been created for both train and test dataset and both contain 99% of sparsity.

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 4, ISSN No. 2455-2143, Pages 223-226
Published Online August 2020 in IJEAST (http://www.ijeast.com)

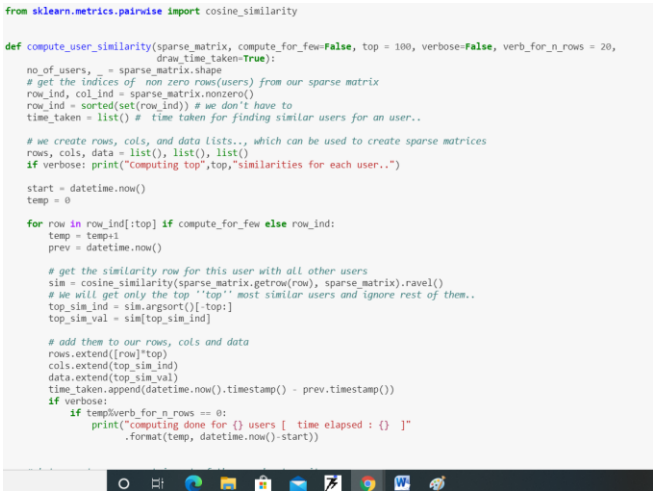


```
start = datetime.now()
if os.path.isfile('train_sparse_matrix.npz'):
    print("It is present in your pwd, getting it from disk....")
    # just get it from the disk instead of computing it
    train_sparse_matrix = sparse.load_npz('train_sparse_matrix.npz')
    print("DONE..")
else:
    print("We are creating sparse_matrix from the dataframe..")
    # create sparse_matrix and store it for after usage.
    # csr_matrix(data_values, (row_index, col_index), shape_of_matrix)
    # It should be in such a way that, MATRIX[row, col] = data
    train_sparse_matrix = sparse.csr_matrix((train_df.rating.values, (train_df.user.values,
                                             train_df.movie.values)),)

    print('Done. It\'s shape is : (user, movie) : ',train_sparse_matrix.shape)
    print('Saving it into disk for furthur usage..')
    # save it into disk
    sparse.save_npz("train_sparse_matrix.npz", train_sparse_matrix)
    print('Done..\n')

print(datetime.now() - start)
```

Fig. 3: calculating sparsity of matrix

4. Computing User-User Similarity matrix

User-User similarity has been calculated 17K dimension for per user and it had been calculated with the help of cosine similarity. Cosine similarity helps to find most similar user to the active user .it finds top similar users and ignore rest of them.

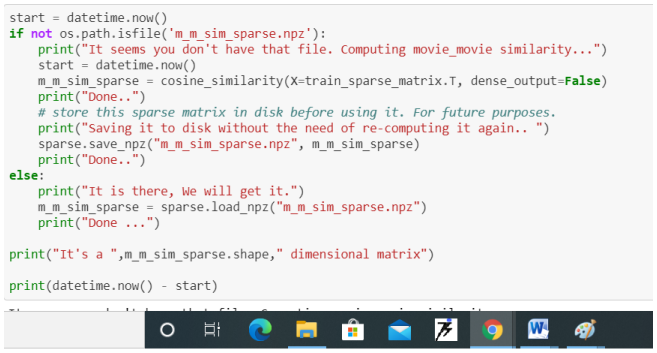


```
from sklearn.metrics.pairwise import cosine_similarity

def compute_user_similarity(sparse_matrix, compute_for_few=False, top = 100, verbose=False, verb_for_n_rows = 20,
                            draw_time_taken=True):
    no_of_users, _ = sparse_matrix.shape
    # get the indices of  non zero rows(users) from our sparse matrix
    row_ind, col_ind = sparse_matrix.nonzero()
    row_ind = sorted(set(row_ind)) # we don't have to
    time_taken = list() #  time taken for finding similar users for an user..

    # we create rows, cols, and data lists.., which can be used to create sparse matrices
    rows, cols, data = list(), list(), list()
    if verbose: print("Computing top",top,"similarities for each user..")

    start = datetime.now()
    temp = 0

    for row in row_ind[:top] if compute_for_few else row_ind:
        temp = temp+1
        prev = datetime.now()

        # get the similarity row for this user with all other users
        sim = cosine_similarity(sparse_matrix.getrow(row), sparse_matrix).ravel()
        # We will get only the top '`top`' most similar users and ignore rest of them..
        top_sim_ind = sim.argsort()[-top:]
        top_sim_val = sim[top_sim_ind]

        # add them to our rows, cols and data
        rows.extend([row]*top)
        cols.extend(top_sim_ind)
        data.extend(top_sim_val)
        time_taken.append(datetime.now().timestamp() - prev.timestamp())
        if verbose:
            if temp%verb_for_n_rows == 0:
                print("computing done for {} users [  time elapsed : {}  ]"
                      .format(temp, datetime.now()-start))
```

Fig. 4: User-User Similarity calculation

5. Computing Movie – Movie Similarity matrix

Even though we have similarity measure of each movie, with all other movies, we generally don't care much about least similar movies. Most of the times, only top x similar items matters. It may be 10 or 100. We take only those top similar movie ratings and store them in a separate dictionary.

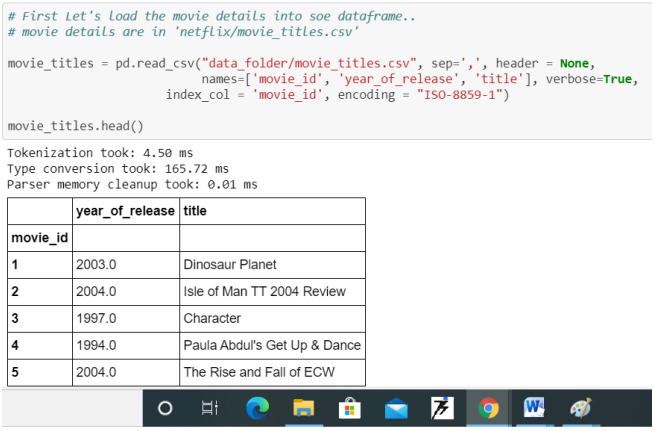


```
start = datetime.now()
if not os.path.isfile('m_m_sim_sparse.npz'):
    print("It seems you don't have that file. Computing movie_movie similarity...")
    start = datetime.now()
    m_m_sim_sparse = cosine_similarity(X=train_sparse_matrix.T, dense_output=False)
    print("Done..")
    # store this sparse matrix in disk before using it. For future purposes.
    print("Saving it to disk without the need of re-computing it again.. ")
    sparse.save_npz("m_m_sim_sparse.npz", m_m_sim_sparse)
    print("Done..")
else:
    print("It is there, We will get it.")
    m_m_sim_sparse = sparse.load_npz("m_m_sim_sparse.npz")
    print("Done ...")

print("It's a ",m_m_sim_sparse.shape," dimensional matrix")

print(datetime.now() - start)
```

Fig. 5: Movie –Movie similarity calculations

6. Finding most similar movies using similarity matrix

From below, with the help of random movie similarities of movies is been checked. If we want to check for particular movie then it also been checked. And for the particular movie how many users has been rated we can see and also we get similar movies with the help of genres and that genres may be of type comedy, Thriller SCI-FI, animated, romantic.so such types of genres provides well recommendations to the users.



```
# First Let's load the movie details into soe dataframe..
# movie details are in 'netflix/movie_titles.csv'

movie_titles = pd.read_csv("data_folder/movie_titles.csv", sep=',', header = None,
                           names=['movie_id', 'year_of_release', 'title'], verbose=True,
                           index_col = 'movie_id', encoding = "ISO-8859-1")

movie_titles.head()
```

```
Tokenization took: 4.50 ms
Type conversion took: 165.72 ms
Parser memory cleanup took: 0.01 ms
```

movie_id	year_of_release	title
1	2003.0	Dinosaur Planet
2	2004.0	Isle of Man TT 2004 Review
3	1997.0	Character
4	1994.0	Paula Abdul's Get Up & Dance
5	2004.0	The Rise and Fall of ECW

Fig. 6: Finding most similar movies

7. Recommendation of top 10 similar movies

This result shows recommendation of 10 movies with movie id, year of its release and its title.

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 4, ISSN No. 2455-2143, Pages 223-226
Published Online August 2020 in IJEAST (http://www.ijeast.com)

Top 10 similar movies

```
movie_titles.loc[sim_indices[:10]]
```

movie_id	year_of_release	title
323	1999.0	Modern Vampires
4044	1998.0	Subspecies 4: Bloodstorm
1688	1993.0	To Sleep With a Vampire
13962	2001.0	Dracula: The Dark Prince
12053	1993.0	Dracula Rising
16279	2002.0	Vampires: Los Muertos
4667	1996.0	Vampirella
1900	1997.0	Club Vampire
13873	2001.0	The Breed
15867	2003.0	Dracula II: Ascension

Fig. 7: Results

IV. CONCLUSION

In the era of big data, RS helps users spend less time finding their favourite items. In the paper, solving data sparsity and high dimensionality, summarize the approaches and techniques of the traditional CF-based recommender systems, and discuss the major challenges and the advantages of the CF-based RS. .The main approach with SVD that it works for good recommendation to users and handle with the problem that user faces like sparsity, scalability, and Dimensionality reduction.

V. REFERENCE

1. Raghuwanshi S., and Pateriya R.(2019). "Collaborative Filtering Techniques in Recommendation System," (pp. 1–21).
2. Thakkar P., Varma K., and Tanwar S. (2019). "Combining User-Based and Item-Based Collaborative Filtering Using Machine Learning,"*Information and Communication Technology for Intelligent Systems*, Smart Innovation, Systems and Technologies 107, (pp. 173–180).
3. Wei J., He J., Chen K., Zhou Y., and Tang Z. (2017). "Collaborative filtering and deep learning based recommendation system for cold start items," *Expert Syst. Appl.*, vol. 69, (pp. 1339–1351).
4. Adomavicius G. and Kwon Y. (2012). "Diversity Using Ranking-Based Techniques," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 5, (pp. 896–911).
5. Ranjan A.A, Rai A., and Haque S. (2019). "An Approach for Netflix Recommendation System using Singular Value Decomposition," *J. Comput. Math. Sci.*, vol. 10, no. 4, (pp. 774–779).
6. McSherry F. and Mironov I. (2009). "Differentially private recommender systems: Building privacy into the netflix prize contenders," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, (pp. 627–635).
7. Isinkaye F., Folajimi Y. and Ojokoh B. (2015). "Recommendation Systems: Principles, methods and evaluation,"*Cairo University*, (pp. 261-273).
8. Koren Y., Bell R. and Volinsky C.(2009). "Matrix Factorization Techniques for Recommendation System,"*IEEE Computer Society*, (pp. 30–37).
9. Ricci F., Rokach L., and Shapira B. (2010). "Introduction to Recommender System Handbook ,"*Egypt. Informatics J.*, vol. 16, no. 3, (pp. 1–35).
10. Lin C.Y., Wang L.C., and Tsai K.H. (2018). "Hybrid Real-Time Matrix Factorization for Implicit Feedback Recommendation Systems," *IEEE Access*, vol. 6, (pp. 21369–21380).
11. Pennington J., Socher R., Manning C. (2014). "Glove: Global Vectors for Word Representation, *"Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 1532–1543).
12. Yuan et al.(2018) "Singular value decomposition based recommendation using imputed data,"*Elsevier*, (pp. 1–10).
13. Ben-Shimon D., Rokach L. and shapira et al.(2016). "An ensemble method for top-N recommendation from the SVD,"*Elsevier*, (pp. 1–9).
14. Guo G., Zhang J. and Yorke-Smith N.(2016). "A Novel Recommendation Model Regularizaed wit User Trust and Item Ratings,"*IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, vol. 28, (pp. 1–14).
15. Barman K., Dabeer O.(2012). "Analysis of a Collaborative Filter Based on Popularity Amongst Neighbours,"*IEEE TRANSACTIONS ON INFORMATION THEORY*, vol. 58, (pp. 1–25).