2021 9th International Conference on Orange Technology (ICOT) | 978-1-6654-7842-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICOT54518.2021.9680609

Movie Recommendation System Using Deep Learning

1st Lakshit Sama, Hua Wang
Institute of Sustainable Industries and Liveable Cities
Victoria University
Melbourne, Australia
lakshit.sama@gmail.com
hua.wang@vu.edu.au

2nd Aaisha Makkar
Postdoctoral Researcher
Seoul National University of Science and Technology
Seoul, South Korea
aaisha.makkar@ieee.org

[1] ***Abstract*—It is a challenge to design a movie recommendation system with artificial intelligent technology. It is difficult to get satisfactory results as not all data is reliable and valuable from the internet. To solve this problem, researchers recommend using a recommendation system which presents relevant information instead of searching the same information multiple times. Generally, collaborative filtering methods based on user information (such as gender type, geographical area, or preference) are effective. However, in today's era, everyone is concerned about suggestions or guidelines so that he/she can make the decision, our area of research includes movies recommendation system by which we can provide the list of similar movies to the user. Now the user is assisted with some lists of movies recommended to the user by our model, he/she only has to decide which movie he should watch next. Instead of looking into the whole list it's better to have some recommendations based on which he/she can decide easily. The overall performance for the purposed framework resulted in an accuracy of 66% which is good for such initiative.**

***Index Terms*—Recommendation, Deep learning, model, accuracy.**

I. Introduction

These days, instead of wasting time on the problem 'what I Should watch next', people prefer to get some recommendations based on his/her previous histories/list of watched movies. By providing them recommendation we assist them so that they can find similar movies on priority bases. Now for them, it saves their great amount of time which they were going to waste to choose by themselves. To recommend we consider the fact if some people watched the same movie and after that which movies do rest of the people watched so that we can predict that he/she might be interested in some movies which other people watched after watching his movie, based on this we provide the user the list of movies similar to the previous one. Deep learning has played as an important approach in various fields such as data analysis in website and health related areas [S. Subramani, et al(2018)], [H. Hu, et al(2006)], [F. Khalil, et al(2007)]. We use deep learning method to anaylse movie recommendation problem. One factor is considered that the rating we choose top rated movies on priority bases and recommend to users.

[1]978-1-6654-7842-7/21/$31.00 ©2021 IEEE

Deep Learning is an integral part of AI used widely for designing algorithms based on the data trend and historical relationship between data [X. Sun, et al(2011)], [H. Wang and L. Sun(2010)], [F. Khalil, et al(2006)], [X. Sun, et al(2008)]. The contributions in this paper are:

1) One dataset is preprocessed by the procedure of feature engineering.

2) The proposed scheme will recommend some other movies to users with KNN clustering methods [J. Huang, et al(2015)], [C. Ke, et al(2017)], [H. Wang, et al(2014)].

3) This scheme has been validated using a deep learning model (Matrix factorization) [G. Bargshady, et al(2020)], [Z. Chen, et al(2019)].

II. Literature Survey

Traditional linear algebra is the foundation of machine learning [D. Pandey, et al(2017)], [W. Gao, et al(2019)], [M. Peng, et al(2018)]. Many machine learning applications are not applied in Recommendation as datasets without NaNs (incomplete en-tries). For example, when building a model, NaN or missing data is trimmed out during data preprocessing because most functions cannot use unpopulated values. If values are missing, features such as principal component analysis are not defined. However, if you get rid of NaN, the recommendation system will not work properly. There are reasons for these NaNs: not every user rate every item, and expecting them to do so is a bit nonsense. Dealing with sparse data can be quite different that's what makes Recommendation an interesting problem [J. Zhang, et al(2015)], [E. Kabir, et al(2020)]. Sparsity complicates things. If no terms are defined in the matrix, the analysis of single value (that is, the factorization of the singular and orthogonal values of the m x n matrix) becomes uncertain. This means that we cannot work out a matrix so explicitly that we can discover which diagonal (or potential) factors in the data have the greatest weight.

Simple yet quite useful are Non-Personalized Recommendations. Due to the fact that they resolve the customer's cold start issue, we can offer some recommendations to the user without knowing anything about them. After obtaining user feedback or acquiring additional information about the user, we can proceed to some of the more advanced models listed below. The methodology provided by IMDB was utilised in the project to determine the top movies in various genres, which can be suggested to any new user. This recommender examines all users who have viewed a specific movie and then counts the number of times the most popular movie from that group is returned. Analogies between movies Without regard for context (purely based on ratings), we utilise the KNN algorithm to discover related movies based on user ratings for various movies. The k-nearest neighbours (KNN) technique is a simple and uncomplicated supervised machine learning approach that may be used to solve classification and regression predicting issues. It is, however, more likely to be utilised in industrial categorisation difficulties. We normally consider three critical aspects while evaluating any technique: [Wang and Wang(2007)]

1. Ease to interpret the output; 2. Calculation time; and 3. Predictive Power.

Matrix Factorizations (Collaborative Filtering) techniques were attempted; the low-rank method is extremely slow, thus the sparse matrix was factored using scipy's SVD. Non-negative matrix factorization (NMF) is a recently discovered technique for identifying linear representations of non-negative data based on its constituents. While this technique has been successfully used to a wide variety of applications, it does not necessarily result in a part-based representation. One of the most advantageous properties of NMF is that it typically delivers data in a sparse representation. This type of representation employs a few 'active' components to encode a large amount of data, allowing for easy interpretation of the coding. [Lee and Seung(2001)] Matrix factorization is a prominent method for developing recommendation systems. It is based on the premise that we can learn users' and films' low-dimensional representations (embedding). For instance, we can determine the number of activities required and the duration of each movie. We may code how long each user enjoys the action, or how long they like the movie, and so on. As a result, we may utilise a combination of user ratings and built-in movies to predict the ratings of unreleased movies. This approach can alternatively be thought of as the following: given a matrix (A [M X N]) containing users and movies, we wish to estimate low-dimensional matrices (W [M X k] and H [M X k]) such that:

$$A = W.HT$$

A. Future Direction

There is an area under deep learning required and worth further exploration. For now, we consider some main directions concerning data, objective, techniques, explanation, and security [R. Rasool, et al(2019)], [Y. Wang, et al(2016)], [Le Sun, et al(2015)]. Project recommendation, and the actual recommendation system, require a multi-objective evaluation, such as accuracy, diversity and even contingency, even the efficiency and quality of online networking services for large-scale projects. Therefore, it is very motivating to use auxiliary data to design more complicated objective functions with different evaluation indicators.

III. PROBLEM FORMULATION

The purpose of the project is to construct software that can be used to recommend movies to the user. People suffer too much to decide what they should watch next, it takes too much of their time to decide, it will be easy for them if some recommendation system does this task and they are left with only some set of movies from which they need to choose their next to be watched. Based on their previous interest our model will recommend them the movies.

The proposed research is aimed to carry out work leading to the development of an approach for Prediction using deep learning the proposed aim will be achieved by dividing the work into the following objectives:

1. Obtain a good dataset (Data collection);

2. Apply deep learning methods to get high accuracy/better results;

3. Validate scheme using deep learning and cross-check the outcomes.

1) Data Set Used: The data set used for this notebook is the 1M rating data set from Movie Lens. This contains 1M ratings of movies from 7120 movies and 14,025 Users. This data set includes:

Movie ID: Each movie is given a unique Id called movie ID, this ID can be used for identification purpose of any movie. Also, any methodologies we are going to implement will be implemented by taking the reference of movie ID only because Movie ID is the identification of any movie and it is unique for each movie.

User ID: Each user is having his/her unique ID so that we can refer users by their ID if in two tests set the same user ID is mentioned it means both are the same user.

Rating: Rating is a score of a movie out of 10, rating depends on various factors like- feedback given by the user, budget and profit of a particular movie. On the bases of rating, we decide whether the movie is hit or flop on Box Office. Besides, a data set of movies includes the movie title, tags, and genres.

Title: Title is the name of the movie, in some cases title may be the same but identification if the movie is done by Movie ID. The title of the movie can explain that 'movie is all about what' to a great extent.

Genres: It explains the type of movie whether it is romantic, action, Drama, Thrill or Adventure, etc. similar type of movie we have to recommend so that Genre plays an important role in the recommendation system.

Tags: Using tags producer or director indicate that what is special about this movie for example- any actor name can be in tags or genre can be in tags, any quote can also be a tag.

IV. METHODOLOGY

- Imported required libraries, the library contains predefined functions which are important for our scheme to implement successfully.

- We need to read the CSV file, consider the require column from CSV file in case one user rated any single movie twice or more then we consider maximum rating it helps in increasing accuracy.
- Visualize the data and classify the rating on the bases of stats given in the dataset and find the percentage of each rating for better understanding.
- Fetched dataset for genres and converted genres in a list so that we can easily implement a matrix factorization scheme.
- Consider the concept of weighted rating score based on the count of reviews def weighted_rating(x,m=min_reviews,C=avg_rating_all):

  $$v = x['count']$$

  $$R = x['mean']$$

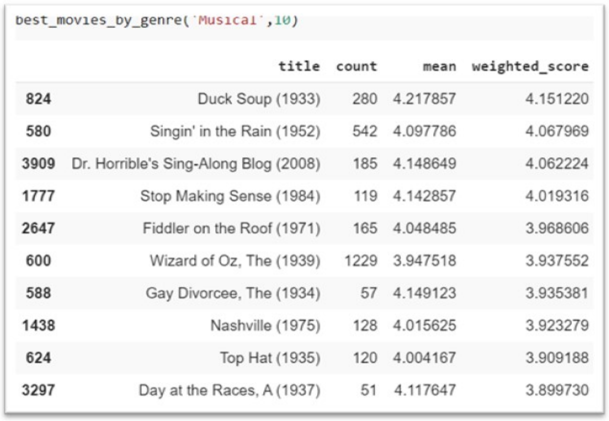
  # Calculation based on the IMDB formula

  $$return(v/(v+m)*R)+(m/(m+v)*C)$$

  Calculate weighted score for each movie.
- Apply join() accordingly for example: join movie score with movie rating.
- Get the best movies according to genre-based on the weighted score which is calculated using IMDB formula, now we have a list of top movies for each genre we can suggest to them and these are important for our recommendation system.
- Created a data frame that has user ratings across all movies in form of matrix used in matrix factorization, whichever movies a user rated in that column there exist a value rest all are ‘NaN’.
- Apply low-rank matrix factorization to find the latent features.
- Merged rating and movie data frame using pd.merge().
- Extracted other top 10 movies which are watched by the people who saw this particular movie, for example: Getting other top 10 movies which are watched by the people who saw ’Gone Girl’ get_other_movies(’Gone Girl (2014)’)
- Created a matrix table with movie IDs on the rows and user IDs in the columns and replace NAN values with 0.
- Gets the top 10 nearest neighbors of any movie using ‘kNN’ (k-nearest neighbors).
- Gets the top 10 movies based on the content of any particular movie using tags and other features.
- Calculated the rmse score of SVD using different values of k (la-tent features).
- Applied matrix factorization scheme which does recommendation based on the above-mentioned features.
- Returned a neural network model that performs matrix factorization.
- Analyzed the recommendation and observe accuracy, accuracy plays an important role.

V. RESULTS

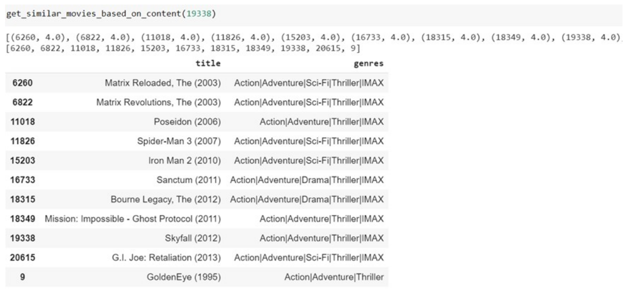
- By implementing deep learning we are able to classify movies according to their genre, let’s say we want to extract top movies for any particular genre then we need to call best_movie_by_genre(‘genre’, List size)



```
best_movies_by_genre('Musical',10)
```

	title	count	mean	weighted_score
824	Duck Soup (1933)	280	4.217857	4.151220
580	Singin' in the Rain (1952)	542	4.097786	4.067969
3909	Dr. Horrible's Sing-Along Blog (2008)	185	4.148649	4.062224
1777	Stop Making Sense (1984)	119	4.142857	4.019316
2647	Fiddler on the Roof (1971)	165	4.048485	3.968606
600	Wizard of Oz, The (1939)	1229	3.947518	3.937552
588	Gay Divorcee, The (1934)	57	4.149123	3.935381
1438	Nashville (1975)	128	4.015625	3.923279
624	Top Hat (1935)	120	4.004167	3.909188
3297	Day at the Races, A (1937)	51	4.117647	3.899730

- We can get the list of movies which are similar to the movie we decide to apply filter and the list of similar movies based on the content and specific genres..

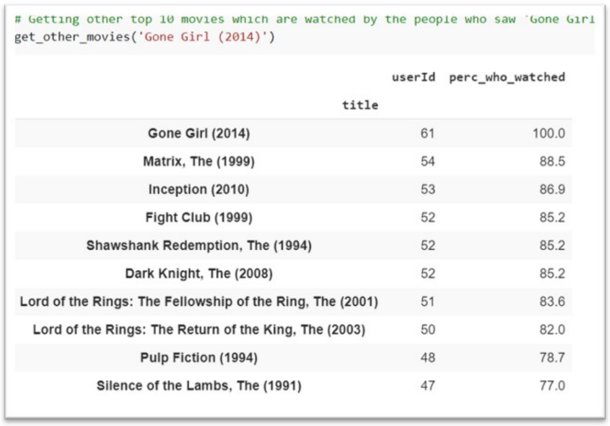


```
get_similar_movies_based_on_content(19338)

[(6260, 4.0), (6822, 4.0), (11018, 4.0), (11826, 4.0), (15203, 4.0), (16733, 4.0), (18315, 4.0), (18349, 4.0), (19338, 4.0)
[6260, 6822, 11018, 11826, 15203, 16733, 18315, 18349, 19338, 20615, 9]
```

	title	genres
6260	Matrix Reloaded, The (2003)	Action|Adventure|Sci-Fi|Thriller|IMAX
6822	Matrix Revolutions, The (2003)	Action|Adventure|Sci-Fi|Thriller|IMAX
11018	Poseidon (2006)	Action|Adventure|Thriller|IMAX
11826	Spider-Man 3 (2007)	Action|Adventure|Sci-Fi|Thriller|IMAX
15203	Iron Man 2 (2010)	Action|Adventure|Sci-Fi|Thriller|IMAX
16733	Sanctum (2011)	Action|Adventure|Drama|Thriller|IMAX
18315	Bourne Legacy, The (2012)	Action|Adventure|Drama|Thriller|IMAX
18349	Mission: Impossible - Ghost Protocol (2011)	Action|Adventure|Thriller|IMAX
19338	Skyfall (2012)	Action|Adventure|Thriller|IMAX
20615	G.I. Joe: Retaliation (2013)	Action|Adventure|Sci-Fi|Thriller|IMAX
9	GoldenEye (1995)	Action|Adventure|Thriller

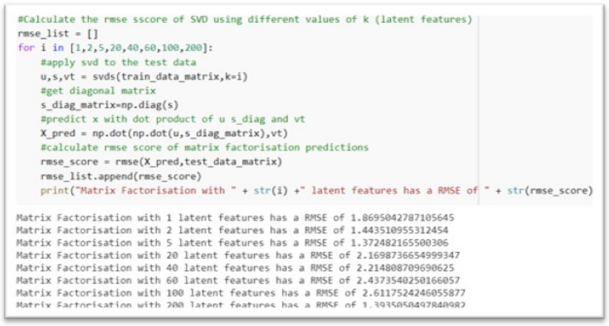
- For example we desire to get similar movies as ‘Gone Girl’ so our model provides this facility too. By using this model we can also get similar movies according to con-tent of the previous movie.



```
# Getting other top 10 movies which are watched by the people who saw 'Gone Girl
get_other_movies('Gone Girl (2014)')
```

	userId	perc_who_watched
title		
Gone Girl (2014)	61	100.0
Matrix, The (1999)	54	88.5
Inception (2010)	53	86.9
Fight Club (1999)	52	85.2
Shawshank Redemption, The (1994)	52	85.2
Dark Knight, The (2008)	52	85.2
Lord of the Rings: The Fellowship of the Ring, The (2001)	51	83.6
Lord of the Rings: The Return of the King, The (2003)	50	82.0
Pulp Fiction (1994)	48	78.7
Silence of the Lambs, The (1991)	47	77.0

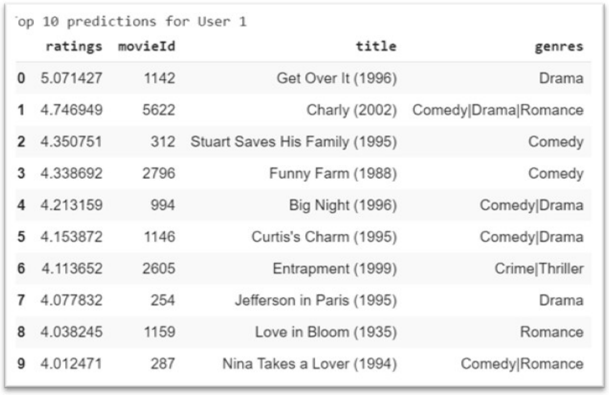
- By calculating rmse score of SVD using different value of k, this is used in matrix factorization and further used in order to get list of movies predicted by movie recommend system.



```
#Calculate the rmse sscore of SVD using different values of k (latent features)
rmse_list = []
for i in [1,2,5,20,40,60,100,200]:
    #apply svd to the test data
    u,s,vt = svds(train_data_matrix,k=i)
    #get diagonal matrix
    s_diag_matrix=np.diag(s)
    #predict x with dot product of u s_diag and vt
    X_pred = np.dot(np.dot(u,s_diag_matrix),vt)
    #calculate rmse score of matrix factorisation predictions
    rmse_score = rmse(X_pred,test_data_matrix)
    rmse_list.append(rmse_score)
    print("Matrix Factorisation with " + str(i) +" latent features has a RMSE of " + str(rmse_score)
```

```
Matrix Factorisation with 1 latent features has a RMSE of 1.8695042787105645
Matrix Factorisation with 2 latent features has a RMSE of 1.443510955312454
Matrix Factorisation with 5 latent features has a RMSE of 1.372482165500306
Matrix Factorisation with 20 latent features has a RMSE of 2.1698736654999347
Matrix Factorisation with 40 latent features has a RMSE of 2.214808709690625
Matrix Factorisation with 60 latent features has a RMSE of 2.4373540250166057
Matrix Factorisation with 100 latent features has a RMSE of 2.6117524246055877
Matrix Factorisation with 200 latent features has a RMSE of 1.3935050497840982
```

- Then we can recommend user with list of movies.



op 10 predictions for User 1

	ratings	movieId	title	genres
0	5.071427	1142	Get Over It (1996)	Drama
1	4.746949	5622	Charly (2002)	Comedy|Drama|Romance
2	4.350751	312	Stuart Saves His Family (1995)	Comedy
3	4.338692	2796	Funny Farm (1988)	Comedy
4	4.213159	994	Big Night (1996)	Comedy|Drama
5	4.153872	1146	Curtis's Charm (1995)	Comedy|Drama
6	4.113652	2605	Entrapment (1999)	Crime|Thriller
7	4.077832	254	Jefferson in Paris (1995)	Drama
8	4.038245	1159	Love in Bloom (1935)	Romance
9	4.012471	287	Nina Takes a Lover (1994)	Comedy|Romance

- This recommendation system is 65.8% accurate, this is impressive as there is no ideal recommendation system exist so this accuracy is quite impressive for such initiative.

VI. CONCLUSION AND FUTURE SCOPE

In general, each strategy can be applied to any kind of help data, although different strategies can result in different efficiencies and efficiencies. To fully utilize the complementarity of different strategies, we believe that designing deep learning strategies will in most cases deliver better performance. Used for collaborative recommendation with deep learning help data, including "knn and matrix decomposition" algorithm. Finally, so far, Knn and matrix factorization is successfully implemented on the schema and the result is quite satisfactory and effective. In the future, deep learning techniques will be widely used to solve other challenging applications, such as video classification, social network analysis, image classification to a whole new level, and logical inferences [Pan and Yang(2009)]. Also, we might provide our scheme as a product or service to the firm which requires an effective recommendation system.

REFERENCES

[S. Subramani, et al(2018)] S. Subramani, et al, "Domestic violence crisis identification from facebook posts based on deep learning," *IEEE access*, vol. 6, pp. 54 075–54 085, 2018.

[H. Hu, et al(2006)] H. Hu, et al, "Combined gene selection methods for microarray data analysis," in *International conference on knowledge-based and intelligent information and engineering systems*, 2006, pp. 976–983.

[F. Khalil, et al(2007)] F. Khalil, et al, "Integrating markov model with clustering for predicting web page accesses," in *AusWeb07*, 2007, pp. 63–74.

[X. Sun, et al(2011)] X. Sun, et al, "Publishing anonymous survey rating data," *Data Mining and Knowledge Discovery*, vol. 23, no. 3, pp. 379–406, 2011.

[H. Wang and L. Sun(2010)] H. Wang and L. Sun, "Trust-involved access control in collaborative open social networks," in *2010 Fourth International Conference on Network and System Security*. IEEE, 2010, pp. 239–246.

[F. Khalil, et al(2006)] F. Khalil, et al, "A framework of combining markov model with association rules for predicting web page accesses," in *Proceedings of the 5th Australasian Data Mining Conference (AusDM 2006): Data Mining and Analytics*, 2006, pp. 177–184.

[X. Sun, et al(2008)] X. Sun, et al, "An efficient hash-based algorithm for minimal k-anonymity," in *CRPIT*, vol. 74, 2008, pp. 101–107.

[J. Huang, et al(2015)] J. Huang, et al, "Topic detection from large scale of microblog stream with high utility pattern clustering," in *Proceedings of the 8th Workshop on Ph. D. Workshop in Information and Knowledge Management*, 2015, pp. 3–10.

[C. Ke, et al(2017)] C. Ke, et al, "Secure k-nn query on encrypted cloud data with multiple keys," *IEEE Transactions on Big Data*, 2017.

[H. Wang, et al(2014)] H. Wang, et al, "Building access control policy model for privacy preserving and testing policy conflicting problems," *Journal of Computer and System Sciences*, vol. 80, no. 8, pp. 1493–1503, 2014.

[G. Bargshady, et al(2020)] G. Bargshady, et al, "Enhanced deep learning algorithm development to detect pain intensity from facial expression images," *Expert Systems with Applications*, vol. 149, p. 113305, 2020.

[Z. Chen, et al(2019)] Z. Chen, et al, "Distributed individuals for multiple peaks: A novel differential evolution for multimodal optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 4, pp. 708–719, 2019.

[D. Pandey, et al(2017)] D. Pandey, et al, "Accurate vessel segmentation using maximum entropy incorporating line detection and phase-preserving denoising," *Computer Vision and Image Understanding*, vol. 155, pp. 162–172, 2017.

[W. Gao, et al(2019)] W. Gao, et al, "Incorporating word embeddings into topic modeling of short text," *Knowledge and Information Systems*, vol. 61, no. 2, pp. 1123–1145, 2019.

[M. Peng, et al(2018)] M. Peng, et al, "Mining event-oriented topics in microblog stream with unsupervised multi-view hierarchical embedding," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 12, no. 3, pp. 1–26, 2018.

[J. Zhang, et al(2015)] J. Zhang, et al, "On efficient and robust anonymization for privacy protection on massive streaming categorical information," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 5, pp. 507–520, 2015.

[E. Kabir, et al(2020)] E. Kabir, et al, "Microaggregation sorting framework for k-anonymity statistical disclosure control in cloud computing," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 408–417, 2020.

[Wang and Wang(2007)] Y. Wang and Z.-O. Wang, "A fast knn algorithm for text categorization," in *2007 International Conference on Machine Learning and Cybernetics*, vol. 6. IEEE, 2007, pp. 3436–3441.

[Lee and Seung(2001)] D. Lee and H. Seung, "Algorithms for non-negative matrix factorization advances in neural information processing 13 (proc. nips 2000)," 2001.

[R. Rasool, et al(2019)] R. Rasool, et al, "Cyberpulse: a machine learning based link flooding attack mitigation system for software defined networks," *IEEE Access*, vol. 7, pp. 34 885–34 899, 2019.

[Y. Wang, et al(2016)] Y. Wang, et al, "Mtmr: Ensuring mapreduce computation integrity with merkle tree-based verifications," *IEEE Transactions on Big Data*, vol. 4, no. 3, pp. 418–431, 2016.

[Le Sun, et al(2015)] Le Sun, et al, "Cloud service description model: an extension of usdl for cloud services," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 354–368, 2015.

[Pan and Yang(2009)] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.