BIL576 PROJECT REPORT

Uğur Güdelek
m.gudelek@etu.edu.tr
Computer Engineering Department
TOBB ETU, Ankara

Simla Burcu Harma
s.harma@etu.edu.tr
Computer Engineering Department
TOBB ETU, Ankara

INTRODUCTION

In today's world, internet has an undeniable part in human life. With the increasing amount of data, reaching the information one needs has become a major research problem. The recommendation systems are useful to reach the desirable data and to increase user's experience, so research has been going on this topic for several decades now. Recommender systems play a big role in the economy of the major companies making use of it, such as Netflix, Amazon.com, Google News, Amazon, Choicestream, etc. Table 1 shows the profit of some websites from the recommendation system [8].

Netflix	2/3rd of the movies watched are recommended
Google News	recommendations generate 38% more click-throughs
Amazon	35% sales from recommendations
Choicestream	28% of the people would buy more music if they found what they liked

Table 1: Companies and their benefit from recommender systems [8].

Movie Recommendation Systems have become popular with the increase in the popularity of movie streaming services like Netflix. Rather than checking the highest rated movies for a genre, users desire to find movies for their special taste.

Recommendation systems are mainly categorized into collaborative filtering, content-based filtering and hybrid recommender systems according to the types of the input. Collaborative filtering systems learn from user-item interactions in the past based on the user ratings or users' browsing history by looking at similar contents in common, the new content recommendation will easily be predicted for each user. This approach has many advantages: it is content-dependent (relies only on connections), as a result of explicit ratings we receive real quality assessment of items, and it provides serendipitous recommendations because it is based on user's similarity [8]. However, as described above, content rates are essential however, not all movies are evaluated by every user in the system so that this causes a problem. Another issue is that diversity between users will be downgraded and within a certain subgroup of users, similar movies are recommended. Collaborative filtering based systems face the cold start problem due to lack of primary information since the user-item matrix can be huge, so sparse. Content-based filtering is based on the user's preference in the past and the item's description. Same aforementioned issues arise when content-based filtering is applied, for instance, if the user's liking is diverse than recommendation will also be diverse. Hybrid approach is another technique combining these two methods.

When applying either methods liking or disliking must be defined in a definitive way. The methods are based on the similarity between users and movies, and this makes the choice of similarity measure important. The vectorized liking and dislikings will be used as inputs to the similarity functions. Several approaches can be found in the literature however more general and widely used similarity metric is the cosine similarity. Thus we will use cosine similarity on our work.

In this project, we work on the question that how can we predict the movies that a person could love, whose favourite movies are known and we will focus on implementing a personalized collaborative filtering movie recommender system. In order to overcome the problems of the approach; we choose our data carefully and as we face other problems on the way, we will adapt and optimize our system. For the experiments, we will use the popular Netflix and MovieLens datasets. For the evaluation of our work, we will use traditional metrics: Mean Absolute Error (MAE) and Root Spare Mean Error (RSME) to measure the error in the predicted ratings and Recall, Precision, Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) to measure the quality of the top-n ranked list of items. In order to implement the movie recommendation system, Python 3 is selected because of the dominance in the area of data processing and machine learning. Python Data Analysis Library (pandas) and Numpy will be examined for data analysis and numerical computation respectively.
We determine Jian Wei et. al.'s timeSVD++ [11] and Rahul Katarya and Om Prakash Verma's movie recommender [6] as our baseline methods since they are recent work and achieve competitive results.

RELATED WORK

Current Recommender Systems in the literature can be classified in three main categories based on the methodology they use: Collaborative Filtering, Content-Based, and Hybrid approaches. Collaborative Filtering approach evaluates similar likes and dislikes on the same content by these two users, and by looking at similar contents in common, the new content recommendation will be predicted for each user. In Content-Based Filtering, a method initially proposed for recommendation of text documents, recommendations are provided for each user just by looking at their previous likes and dislikes. This method has several drawbacks such as the diversity of user's liking will result on diverse recommendations. Hybrid approach, on the other hand, makes use of both techniques and takes their advantageous parts [3, 4].

There has been a significant amount of work on Collaborative Filtering approaches in Recommendation Systems. D.K. Yadav et. al. presented a collaborative-filtering based movie recommendation system: MOVREC, which makes use of the information provided by users [8]. They sorted movies with previous users' ratings by using K-means clustering. They used their own dataset by taking the information from IMDB. To evaluate the success of the algorithm they proposed and their system, they tested their system over a small group of people.

Another work on movie recommenders using traditional user-based Collaborative Filtering uses the pre-filled sparse user project scoring matrix [12]. In order to deal with complex relations in database system they used graph database in their recommender. They used MovieLens100K data sets, including 100,000 ratings from 1000 users on 1700 movies. However they did not mention any experiments.

Since content-based approaches on collaborative filtering suffer from poor accuracy, scalability, data sparsity and big-error prediction [9]; researchers use item-based collaborative filtering approach. They used Netflix user item ratings dataset which was available on Movie Lens web site. The dataset consist of one million ratings of 6040 users over for 3952 movies. In evaluation of their work they used Mean Absolute Error (MAE) and experimented for various neighbor sizes. The lowest MAE is 0.938 when the number of neighbors is 20. Mukesh Kumar Kharita[7] et. al. also used item-based collaborative filtering with the same motivation and the difference of their research was

being in real time, making the system dynamic and trainable from positive use feedback. They used MovieLens dataset with ratings given by 943 users for more than 1682 movies. They state that the accuracy of the system is low compared to the other recommendation systems in the literature, however being in real-time makes their research standout from the rest.

In order to improve the accuracy, KNN is another popular approach on movie recommenders using collaborative filtering. V. Subramaniyaswamy and R. Logesh developed a movie recommender using their proposed Adaptive KNN algorithm for collaborative filtering [10]. They experimented on MovieLens dataset, and evaluated the results with precision, recall, F-measure and accuracy. They overperformed the baseline methods on their research. Bei-Bei CUI also designed a movie recommendation system using KNN algorithm [2].

While collaborative filtering is one of the most popular techniques for recommendation systems, it suffers from cold-start problem. When there is a small number of rating records or no rating records, the cold-start problem arises. Jian Wei et. al. proposed two recommendation models to solve this problem [11]. They used a specific deep neural network SADE to extract the features of the items, and they modified the state of the art model timeSVD++ to take the content features into prediction of ratings for cold start items. They experimented on Netflix rating dataset and outperformed the baseline models. They used Root Mean Squared Error (RMSE) as a metric and obtained 1.048 on 300 movies.

In order to improve the quality of predictions and to overcome cold-start problem, various clustering techniques have been used. K-means algorithm is one of the most popular ones and for collaborative filtering researchers use this algorithm with genetic algorithms. Rahul Katarya and Om Prakash Verma [6] proposed a collaborative recommender system making use of k-means clustering by adopting cuckoo search optimization algorithm. They performed their experiments on MovieLens dataset and used MAE as an evaluation metric. Their system has 0.68 MAE which outperforms previous approaches. Another movie recommender system [5] developed by Rahul Katarya uses k-means clustering algorithm with bio-inspired artificial bee colony (ABD) optimization technique. In this research, for the experiments, again, the MovieLens dataset is used. Precision, MAE, recall and accuracy were used as evaluation metrics for this work. 0.764 MAE is obtained for k=15. Meng-Hui Chen et. al. also used a bio-inspired technique [1]. In their work, they applied artificial immune network to collaborative filtering for movie recommendation. For their experiments they used MovieLens and EachMovie datasets; and for evaluation, they used MAE, precision and recall. They obtained around 0.72 MAE with optimized parameters.

PROPOSED SOLUTION

Collaborative Filtering (CF) approach is the most popular approach in the domain of recommender systems. It uses a large amount of data to utilize item recommendation for individual users while taking the consideration of other user's rating on the same item. However, the main issue in CF is the poor performance on the sparse user-item matrix and poor recommendation results for items which have zero or a few ratings. This problem is known as cold-start (CS) problem. When there is no rating for a specific item, it means that there also be no collaborative information about that particular item. In the same manner, few ratings mean few collaborative information the system can get.

We propose a recommendation system to overcome the CS problem. In order to construct this recommender system, first, Movie items are split into three categories: non-cold-start items (NCS), incomplete-cold-start-items (ICS), complete-cold-start-items (CCS). In detail, NCS stands for the items which have enough rating to apply CF; ICS stands for the items which have few rating means that they have not enough rating to apply proper CF and CCS stands for the items which have no rating means that they cannot transmit any information to CF, therefore no recommendation based on these items can be applied while using classical CF approach. This shows that NCS and ICS items need a sophisticated approach to used in CF based recommender systems.

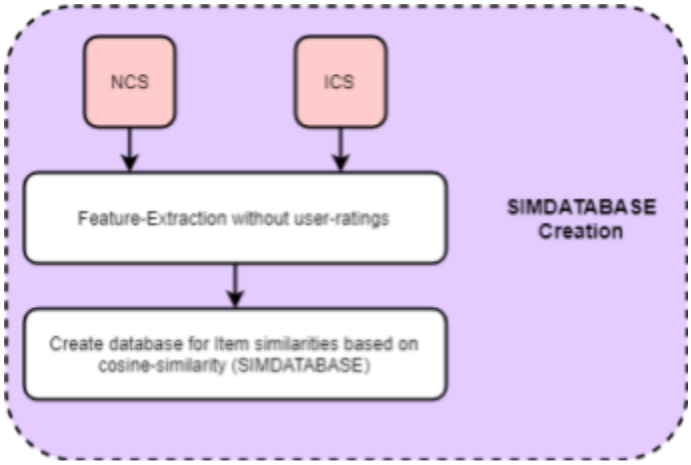

Figure 1. SIMDATABASE creation process

In order to apply this sophisticated approach, first, similarity database (SIMDATABASE) is created with the NCS and ICS items (Figure 1). The database creation process requires movie features such as title, plot, genre, IMDB score, etc. except user ratings. User ratings cannot be used in this step because the actual problem is the lack of enough user ratings. Pre-movie-vectors are constructed using general features about the movie as described above. One feature can be a rating of director or sum of a rating of actor or actress. Pre-movie-vectors represents the points in feature-space. Some similarity metrics can be calculated on these representation points. Therefore, similar items can be clustered using any clustering algorithms such as k-means.

Created SIMDATABASE is used to obtain CCS item. Pre-movie-vectors are retrieved based on the similarity of items in the SIMDATABASE. Similarity metric in this step is selected as cosine-similarity (Equation 1) which is a normalized form of Euclidean distance.

$$similarity = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad \textbf{(1)}$$

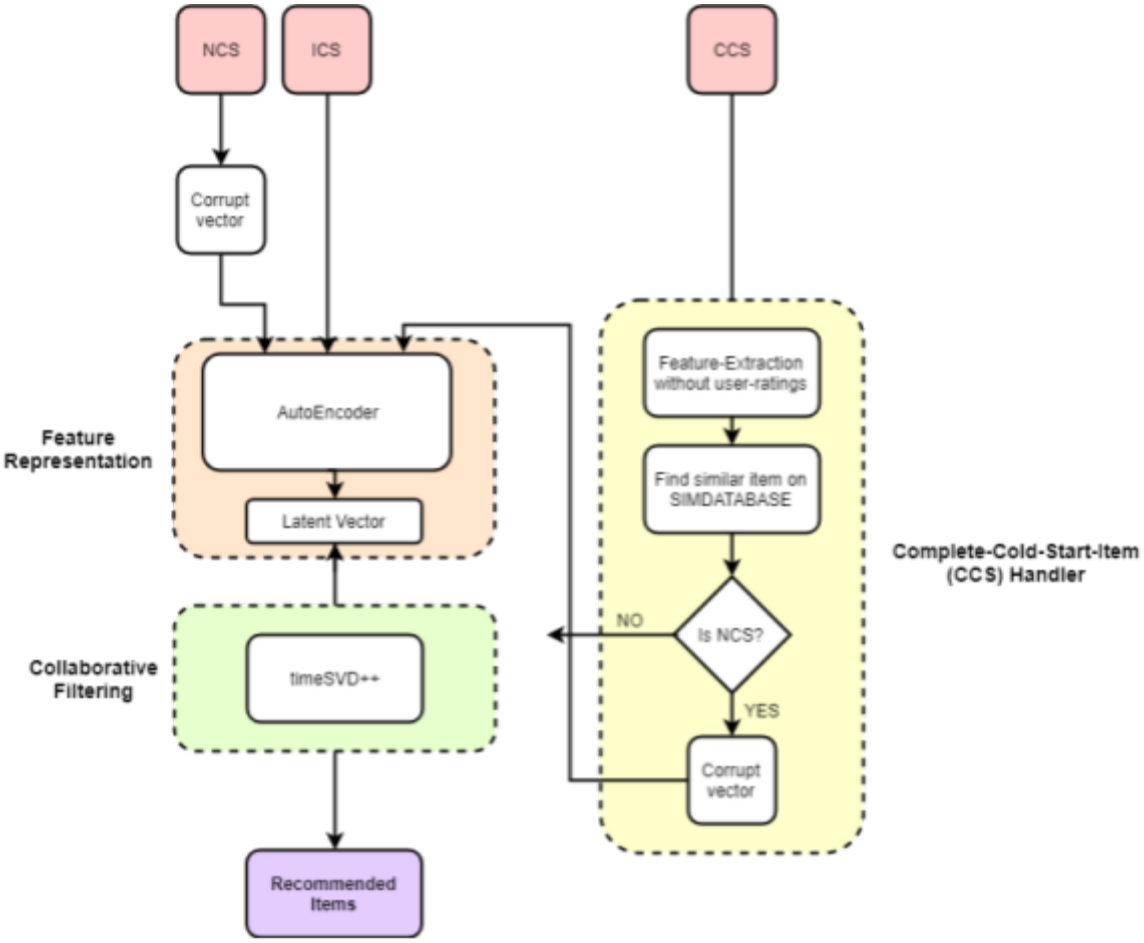

Figure 2. Flowchart of the proposed system

General flowchart of the system can be seen in Figure 2. In this design, Pre-movie-vectors are used for NCS and ICS items, and pre-movie-vectors are retrieved from SIMDATABASE for CCS items. In the system, AutoEncoder deep learning architecture is used to create latent vectors for each item. The AutoEncoder (AE) architecture consists of two main components: encoder, decoder (Figure 3). Encoder layers squeeze the input vector which is pre-movie-vector and decoder layers reverse this process. The output of the last encoder layer is the latent vector which is a dimensionally reduced representation of a large input vector. The AutoEncoder uses this latent vector to create new large output vector which has the same dimensions as the input vector. The reconstruction error while creating the output vector should be minimized to obtain the good latent vector. Same output and input vector means 0 error which means all information in the input vector is represented in dimensionally reduced latent vector.

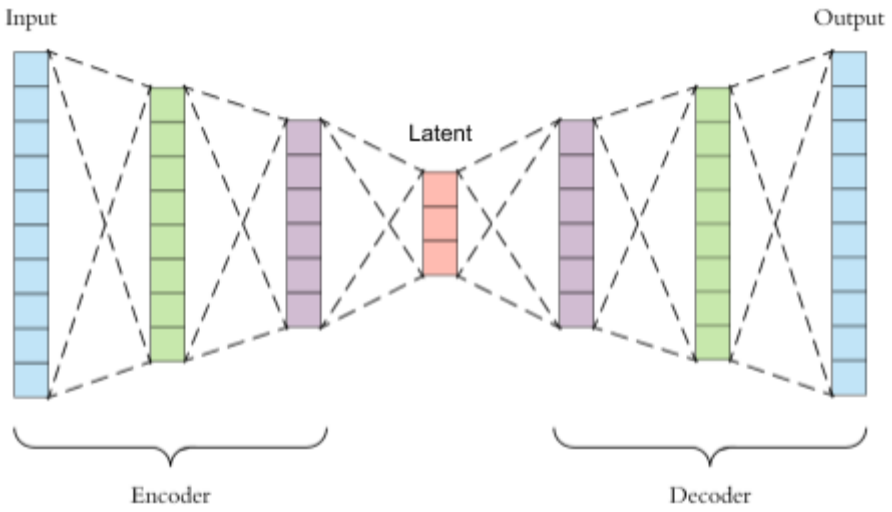

Figure 3. Auto-Encoder Architecture

The output of the AE represents the movie items as a latent vector which is available for NCS, ICS, and CCS, therefore, these vectors can be used in any collaborative filtering approach. In this work, we used the timeSVD++ which is the extension of singular-value-decomposition (SVD - Equation 2). In addition to SVD, timeSVD++ also utilizes the temporal information. timeSVD++ gives, *user-to-concept*(**U**), *concept-strengths*(**Σ**) and *movie-to-concept*(**V**).

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{2}$$

Experimental Evaluation/Preliminary Results

Do the techniques such as Singular Value Decomposition (SVD) can be improved with the help of additional features? More specifically, can utility matrix representation be improved before the decomposition process? Are the improved version of the utility matrix be able to beat the standard approaches? To find an answer to these questions, we conduct an experiment over MovieLens dataset and auto-encoder deep learning approach is used in these experiments to fill sparse utility matrix representation.

We have implemented our experiments with Python3. It is a perfect selection when working with data and machine learning. Numpy library is used for numerical computations, scipy is used for calculating cosine similarities between particular CCS item and all NCS items. Pandas is used for IO operations and data preprocessing. Plotly, matplotlib and seaborn is used to visualize plots of the experiment. Tqdm is useful when the progress bar feature is a necessity. Pytorch and torchvision libraries are used for building, training, and testing of the autoencoder. The surprise is used for training and testing some collaborative filtering algorithm such as SVD, SVD++, KNNBasic.

We study the public MovieLens dataset to conduct the experiments, which have 100k ratings by 943 users and 1682 movies, of scale 1-5. This dataset was collected by the GroupLens Research Project at the University of Minnesota. It has only the users who rated at least 20 movies. It consists of simple movie features i.e. genres, simple demographic info for the users (age, gender, occupation, zip) and more importantly user-movie utility matrix for the collaborative filtering approach.

Table 1. *MovieLens dataset statistics, completely-cold-start items, and non-cold-start items statistics (CCS and NCS are constructed on MovieLens dataset by looking for each items' rating count < 5 or not).*

	MovieLens		
	Total	**CCS items (rating_count < 5)**	**NCS items ((rating_count >= 5))**
Number of users	943	234	943
Number of movies	1682	333	1349
Number of ratings	100k	713	99287
μ(ratings)	3.07	2.64	3.53
μ(rating count)	59.4	2.14	73.6

We have selected root-mean-squared error (RMSE) as an evaluation measure (Equation 3). Most-common collaborative filtering evaluation metric is RMSE and MAE. RMSE computes the mean of residuals, on the other hand, MAE computes the median of residuals. RMSE takes outliers into account more than MAE. Therefore, the results will be worse when some prediction errors are quite high.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad \textbf{(3)}$$

Following set of parameters is used during the experiment:
Dataset parameters; ccs_threshold=5 is selected to split CCS items and NCS items, train_test_size = 0.25 is selected to split train and test dataset to evaluate our overall system. After the split, trainset and testset global means are 3.5296 and 3.5349 respectively, which states that the split process creates sets which have a similar distribution.
Auto-encoder parameters; the architecture consists of 5 layers with 21, 20, 5, 20, 21 neurons on each, respectively. In addition, ReLUs at the end of each inner layer is used to create non-linear architecture. Latent vector dimension size is 5. This means that each movie is represented with the 5-dimensional latent vector. The movie features consist of 21 features: 19 genre features, 1 rating count and 1 rating mean. Respectively, num_epochs and batch_size are 100 and 20. In addition, ADAM optimizer with 1e-3 learning rate is used to train auto-encoder.

Overall algorithm logic is designed as follows:

```
1. Create MovieLens Dataset
2. Split CCS and NCS items (n_ratings < 50)
3. Train AE with movie contents
4. m = pick a CCS item
5. Loop until m is not CCS anymore:
       u = pick random user
       cos_sims = cosine similarities between latent(u.rated_movies) vs latent(m).
       (u,m).rating = mean(cos_sims.top10())    -> fill movielens dataset accordingly.
6. Repeat step 4 until 0 CCS item left.
7. Send (user x movies) utility matrix to surpriselib SVD++
8. Calculate RMSE on test set
```

We determine Jian Wei et. al.'s timeSVD++ [11] and Rahul Katarya and Om Prakash Verma's movie recommender [6] as our baseline methods since they are recent work and achieve competitive results. In addition, they both try to use the cold start problem of collaborative filtering.

Jian Wei et. al.[11] proposed specific deep neural network SDAE to extract the features of the items, and they modified the state of the art model timeSVD++ to take the content features into the prediction of ratings for cold start items which is similar to our approach. They experimented on Netflix rating dataset with OMDb plot database. They used Root Mean Squared Error (RMSE) as a metric and obtained 1.048 on 300 movies. We decide to compare our RMSE and their's on Netflix dataset later because we only conduct our experiments on MovieLens dataset for now.

Rahul Katarya and Om Prakash Verma [6] proposed a collaborative recommender system making use of k-means clustering by adopting cuckoo search optimization algorithm. They performed their experiments on MovieLens dataset which is also our dataset and used RMSE as an evaluation metric. Their system has 1.23104 RMSE which outperforms previous approaches. We will compare our RMSE and their's on MovieLens dataset.

Table 2. *Experimental Results (AE means encoded movie features used for filling utility matrix. RAW means raw utility matrix is used).*

	AE	***RAW***	***KNN + Cuckoo Search***
SVD++	**0.9208**	0.9216	-
SVD	**0.9329**	0.9344	-
KNNBasic	**0.9746**	0.9851	-
Katarya et al. [6]	-	-	1.23104

Table 2 shows that our proposed solution -AE approach- wins every combination of collaborative filtering algorithms such as SVD++, SVD, and KNNBasic. Additionally, the solution of Katarya et al. -KNN + Cuckoo Search- have 1.23 RMSE result which is way behind standard SVD and our solution. In addition to that, Figure 3 shows that except for SVD++, our proposed system surpass raw results. This means that filling the utility matrix with the help of more complex approaches like autoencoder is beneficial for the overall system.

Our designed experiment is only using mean rating and rating count for each item along with genres. These features can be increased with the help of several mixtures of datasets. Therefore, the proposed solution seems to be appealing and further analysis and experiments are required.

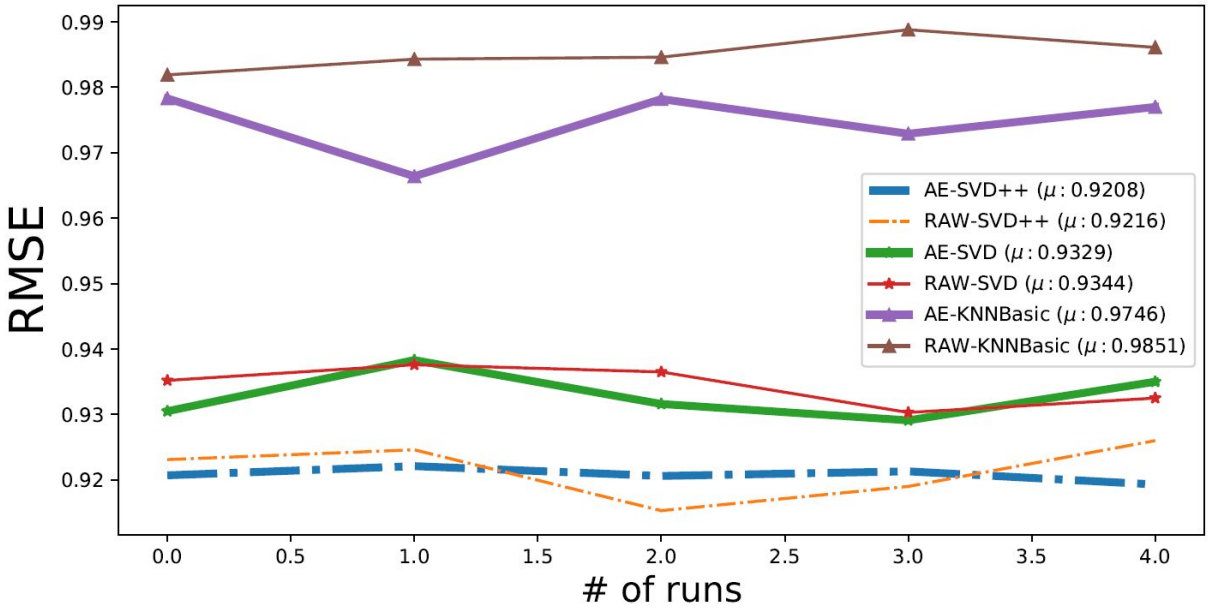

Figure 3. Proposed solution vs raw utility matrix solution with the *comparison of several collaborative filtering approaches*

Current Status: Implementation & Experiments Progress So far

We have done the logic of our experimental design together. Thinking of every tiny bit of the design are quite hard work to do alone. Also, we work together during the implementation process. However, if we need to specify which part of our implementation is done by who; here it is as follows:

Simla: I have implemented the MovieLens dataset class and retrieve its statistic. I have also created the results tables and figures.

Uğur: I have implemented overall logic, AutoEncoder, and collaborative filtering algorithms.

REFERENCES

[1] Chen, Meng-Hui, Chin-Hung Teng, and Pei-Chann Chang. "Applying artificial immune systems to collaborative filtering for movie recommendation." *Advanced Engineering Informatics* 29.4 (2015): 830-839.

[2] Cui, Bei-Bei. "Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm." *ITM Web of Conferences*. Vol. 12. EDP Sciences, 2017.

[3] Geetha, G., et al. "A hybrid approach using collaborative filtering and content based filtering for recommender system." *Journal of Physics: Conference Series*. Vol. 1000. No. 1. IOP Publishing, 2018.

[4] Jung, Kyung-Yong, Dong-Hyun Park, and Jung-Hyun Lee. "Hybrid collaborative filtering and content-based filtering for improved recommender system." *International Conference on Computational Science*. Springer, Berlin, Heidelberg, 2004.

[5] Katarya, Rahul. "Movie recommender system with metaheuristic artificial bee." *Neural Computing and Applications* 30.6 (2018): 1983-1990.

[6] Katarya, Rahul, and Om Prakash Verma. "An effective collaborative movie recommender system with cuckoo search." *Egyptian Informatics Journal* 18.2 (2017): 105-112.

[7] Kharita, Mukesh Kumar, Atul Kumar, and Pardeep Singh. "Item-Based Collaborative Filtering in Movie Recommendation in Real time." *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*. IEEE, 2019.

[8] Kumar, Manoj, et al. "A movie recommender system: Movrec." *International Journal of Computer Applications* 124.3 (2015).

[9] Ponnam, Lakshmi Tharun, et al. "Movie recommender system using item based collaborative filtering technique." *2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS)*. IEEE, 2016.

[10] Subramaniyaswamy, V., and R. Logesh. "Adaptive KNN based recommender system through mining of user preferences." *Wireless Personal Communications* 97.2 (2017): 2229-2247.

[11] Wei, Jian, et al. "Collaborative filtering and deep learning based recommendation system for cold start items." *Expert Systems with Applications* 69 (2017): 29-39.

[12] Yi, Ningning, et al. "Design and implementation of movie recommender system based on graph database." *2017 14th Web Information Systems and Applications Conference (WISA)*. IEEE, 2017.