Research on Context-Aware Recommender Systems

Duc T. Nguyen

November 30, 2022

Abstract

With the explosion of information on the internet, people usually struggle with finding content that is appropriate to them. This is where recommender systems come into play. Recommender systems can understand users' behaviors and give them contents they might want to watch based on their past behaviors. However, traditional recommender systems do not take into account contextual factors such as weather, traffic jam, companion, etcetera. This thesis aims to present traditional recommender system problems as well as a method to integrate contextual factors into the recommendation process and evaluate the result.

1 Traditional Recommender Systems

Traditional Recommender Systems are intelligent systems that can recommend to users a list of items, and products that are closely related to them based on the profile of users and items. In short, the objective of those systems can be modeled as follow:

$$R : Users \times Items \rightarrow Ratings \tag{1}$$

Recommender Systems try to find function R that takes users' profiles ($Users$) and items' profiles ($Items$) as arguments to give output $Ratings$ as predictions of how much users rate items. Then the systems will filter out items that have low predicted ratings to suggest to users:

$$rec(u) = argmax(R(u, i))|\forall u \in Users, \forall i \in Items \tag{2}$$

There are three types of traditional Recommender Systems: Content-based, Collaborative Filtering, and Hybrid Recommender Systems.

1.1 Content-based Recommender System

Content-based Recommender System (CBRS) estimates users' ratings by calculating the similarity of content-based user preferences and content of items [1], then ranking the results to filter out low-ranking items:

$$R(u, i) = sim(ContentBasedProfile(u), Content(i))|\forall u \in Users, \forall i \in Items \tag{3}$$

ContentBasedProfile and *Content* are feature extraction functions that vectorize metadata of new items as well as items that users have already viewed to build users' profiles. For example, the metadata of a movie could be its genre, actors, ratings from trusted sources, description, plot, etcetera. The feature extraction function could be TF-IDF (Term Frequency – Inverse Document Frequency) [2] if metadata is text-based data. Then a user profile can be the mean of the feature vector of items that the user viewed in the past. The similarity function could be a heuristic function such as cosine or Pearson coefficient [3] or a function that is learned by supervised classifiers such as Bayes Classifier [4], Artificial Neural Network [5], etcetera.

As an advantage, CBRS can work well on systems where users rarely rate items (sparse data problem) because CBRS relies on the content of items instead of users' ratings. Additionally, it also works well when there is a new item (cold start problem). On the other hand, CBRS may only suggest the same type of items to users. Besides, if items' metadata does not describe items correctly, the accuracy of CBRS might be affected. Also, contextual information has not been taken into account.

1.2 Collaborative Filtering Recommender System

Collaborative Filtering Recommender System (CFRS) does not take into account the content of items (or products) when recommending to users. Instead, CFRS will try to find the hidden patterns inside the rating matrix of users and items that reflect the users' preferences in the past [1], then give suggestions based on those patterns.

Overall, CFRS can be stated as follow: Given rating matrix $R_{n\times m}$ where n is the number of users and m is the number of items, and r_{ij} is the rating of the i^{th} user toward the j^{th} item. However, there are missing values r_{ij} because the i^{th} user has not rated the j^{th} item yet. Thus, CFRS will predict those values and then rank users' unseen items to suggest to them. The prediction process can be based on two methods: Memory-based and Model-based methods.

1.2.1 Memory-based Collaborative Filtering

Memory-based CFRS focuses on calculating the correlation between users (User-based algorithm) or items (Item-based algorithm) using a heuristic function such as Pearson [3], Cosine correlation, Euclidean distance, mean square deviation, etcetera. The top highest correlation values will be used as weights when predicting missing ratings.

For the User-based algorithm, missing ratings can be calculated as follow:

$$r_{ui} = b_u + \frac{\sum_{v\in U} p_{uv} \cdot (r_{vi} - b_v)}{\sum_{v\in U} |p_{uv}|} \tag{4}$$

r_{ui} and r_{vi} are ratings of user u and v toward item i, b_u and b_v are the mean of ratings of user u and user v respectively, p_{uv} is the correlation between u and v, and U is a collection of users that are the most similar to user u in terms of correlation values and rated item i.

Similarly, the Item-based algorithm has been formulated as follows:

$$r_{ui} = b_i + \frac{\sum_{v\in I} p_{iv} \cdot (r_{uv} - b_v)}{\sum_{v\in I} |p_{iv}|} \tag{5}$$

r_{ui} and r_{uv} are ratings of user u toward items i and v, b_i and b_v are the mean of ratings for items i and v respectively, p_{iv} is the correlation between i and v, and I is a collection of items that are the most similar to item i in terms of correlation values and rated by user u.

One of the popular correlation functions is Pearson correlation [3]. It is essentially a cosine similarity between two vectors after centralizing:

$$p_{ij} = cos((v_i - b_i), (v_j - b_j)) = \frac{(v_i - b_i) \cdot (v_j - b_j)}{||v_i - b_i|| \cdot ||v_j - b_j||} \tag{6}$$

p_{ij} is the Pearson correlation of vectors v_i and v_j, b_i and b_j are baseline vectors and were calculated by $b_i = \frac{\sum_{i\in I} r_i}{||I||}$ where r_i is rating and I is a collection of users or items.

1.2.2 Model-based Collaborative Filtering. SLIM algorithm

Rather than using a heuristic function to estimate the correlation, Model-based CFRS applies machine learning and data mining algorithms to find optimal similarities throughout the data training process. Within the scope of this thesis, the Sparse Linear Method (SLIM) algorithm [6] will be given as an example.

The idea of SLIM is predicting the rating of a user based on his/her past ratings together with weight parameters of the most similar items in terms of Pearson correlation values:

$$\bar{R} = R \cdot W \tag{7}$$

$\bar{R}$ is fully filled rating matrix of n users and m items whereas R is the actual rating matrix with missing values. W is a matrix weight parameter that needs to be learned. This function can also be written

as follows:

$$\bar{r}_{ui} = \sum_{v \in I} r_{uv} \cdot w_{iv} \tag{8}$$

$\bar{r}_{ui}$ and r_{uv} are ratings of user u toward items i and v. I is a collection of items that are the most similar to item v in terms of Pearson correlation. w_{iv} is the similarity (or weight parameter) between items i and v that needs to be learned.

SLIM learns parameter W by optimizing the square loss function with L_1 and L_F regularization [7]:

$$\begin{aligned} Loss &= \frac{1}{2}||R - \bar{R}||_F^2 + \frac{\beta}{2}||W||_F^2 + \alpha||W||_1 \\ &= \frac{1}{2}\sum_{u \in U}\sum_{i \in I}(r_{ui} - \bar{r}_{ui})^2 + \frac{\beta}{2}\sum_{i \in I}\sum_{v \in I} w_{iv}^2 + \alpha\sum_{i \in I}\sum_{v \in I} w_{iv} \end{aligned} \tag{9}$$

The $||R - \bar{R}||_F^2$ reflects the difference between the predicted and actual ratings while L_1 and L_F regularization prevent the overfitting problem [**?**]. α and β are hyperparameters to control how severe the penalty is. SLIM uses the Stochastic Gradient Descent (SGD) algorithm [8] to minimize the loss function to find the optimal W. SGD algorithm is an iterative method for optimizing an objective function by gradually reducing its gradient. At each step, SGD updates parameters so that the value of the loss function reduces over time. In this case, the parameters will be updated at each step as follow:

$$\begin{aligned} w_{iv} &= w_{iv} - \mu \cdot \frac{\partial Loss}{\partial w_{iv}} \\ &= w_{iv} - \mu \cdot ((r_{ui} - \bar{r}_{ui}) \cdot r_{uv} + \beta \cdot w_{iv} + \alpha) \end{aligned} \tag{10}$$

w_{iv} is the weight parameter between items i and v. r_{ui} and r_{uv} are actual ratings of user u toward items i and v whereas $\bar{r}_{ui}$ is the predicted rating toward item i. μ is a hyperparameter that controls the learning speed and α and β are regularization hyperparameters.

1.3 Hybrid Recommender System

Hybrid RS is actually a recommender system that runs both CBRS and CFRS and then combines their results together. With this strategy, the system can overcome the disadvantages of CBRS and CFRS. However, the contextual factors which are very important to users' decisions have not been integrated into recommending process yet. Therefore, this thesis will present some methods to integrate that information to recommendation algorithms in the next section.

2 Context-aware Recommender System

2.1 Introduction

Context is the list of information that describes factors around users when they are using applications such as weather, place, companion, etcetera. There are three approaches to integrating this information [1]:

- *Contextual Pre-Filtering*: Filter out ratings that are not in the same context before running the recommendation process.
- *Contextual Post-Filtering*: Filter out results of the recommendation process based on the user context.
- *Contextual Modelling*: Using context in the data training process instead of using it separately. This thesis will focus on this method to build a context-aware RS.

2.1.1 Independent Contextual Modelling

This method concatenates the rating matrix and context vectors to form new a rating tensor with both rating and context values. The training and the predicting process usually use the matrix factorization method to infer the missing rating values. Tensor Factorization algorithm [9] is a good example of this approach.

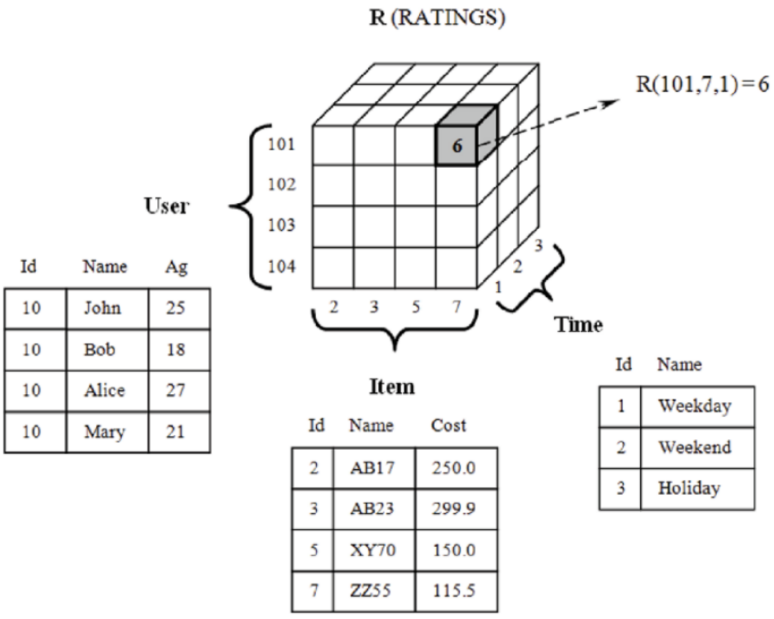

Figure 1: Tensor Factorization ACM RecSys 2010

However, this method also has a drawback in that the relation between context and users or items has not been considered. For example, there are some users who only give high ratings on rainy days or only watch cartoons when being by their kids. This problem can be solved by the Dependent Contextual Modelling approach.

2.1.2 Dependent Contextual Modelling

This approach extracts contextual features at the same time as training the traditional RS model. There are two approaches to do this: *Deviation-based Contextual Modeling* [10] and *Similarity-based Contextual Modeling* [11]. However, [11] proved that the Similarity-based approach can outperform the Deviation-based approach. Thus, this thesis only covers *Similarity-based Contextual Modeling*.

The idea of *Similarity-based Contextual Modeling* is that the prediction will rely on how similar the current context and no context are. It means that if there is no context, the prediction will use the result of traditional RS. Otherwise, the prediction will be affected by the similarity.

$$\bar{R}_{uic} = \bar{R}_{ui} \cdot sim(c, c_\emptyset) \tag{11}$$

$\bar{R}_{uic}$ is then prediction whereas $\bar{R}_{ui}$ is the result of traditional RS. $sim(c, c_\emptyset)$ is the similarity of context c and no context $c_\emptyset$. The similarity can be simply the inverse of Euclidean Distance or other similarity functions:

$$sim(c_k, c_l) = 1 - \sqrt{\sum_{i=1}^{D}(c_{ki} - c_{li})^2} \tag{12}$$

D is the number of context dimensions. $sim(c_k, c_l)$ is the similarity of two context vectors. c_{ki} and c_{li} are the i^{th} dimension of two context vectors. These vectors are also parameters that need to be learned through the data training process.

The next section will present an algorithm that applies this approach to form a Context-Aware Collaborative Filter RS based on the SLIM algorithm.

2.2 Contextual Sparse Linear Method Multidimensional Context Similarity (CSLIM-MCS)

Overall, Contextual Sparse Linear Method Multidimensional Context Similarity (CSLIM-MCS) algorithm integrates contextual factors into the recommendation process of the SLIM [6] algorithm by applying the *Similarity-based Contextual Modeling* technique:

$$\bar{R}_c = R \cdot W \cdot sim(c, c_\emptyset) \tag{13}$$

$\bar{R}_c$ and R are the predicted and actual rating matrices respectively. W is the weight matrix of the SLIM algorithm and $sim(c, c_\emptyset)$ is the similarity of context c and no-context $c_\emptyset$. W, c, and $c_\emptyset$ are parameters that need to be learned in the CSLIM-MCS algorithm. The formula can also be rewritten as follow:

$$\begin{aligned} \bar{r}_{uic} &= \sum_{v \in I} r_{uv} \cdot w_{iv} \cdot sim(c, c_\emptyset) \\ &= \sum_{v \in I} r_{uv} \cdot w_{iv} \cdot (1 - \sqrt{\sum_{k=1}^{D} (c_k - c_{\emptyset k})^2}) \end{aligned} \tag{14}$$

$\bar{r}_{uic}$ is the predicted rating of user u toward item i in context c. r_{uv} is the actual rating of user u toward item v. w_{iv} is the weight parameter of items i and v. I is the collection of items that are similar to item i in terms of Pearson correlation. D is the dimensions of context vectors. c and $c_\emptyset$ are context vectors and also parameters of the CSLIM-MCS algorithm. CSLIM-MCS also uses the Stochastic Gradient Descent (SGD) to minimize square error which is similar to traditional SLIM. However, due to the change in predicting function, the partial derivative of the loss function will also change:

$$\begin{aligned} \frac{\partial Loss}{\partial w_{iv}} &= \frac{\partial(\frac{1}{2}\sum_{u \in U}\sum_{i \in I}(r_{uic} - \bar{r}_{uic})^2)}{\partial \bar{r}_{uic}} \cdot \frac{\partial \bar{r}_{uic}}{\partial w_{iv}} + \frac{\partial(\frac{\beta}{2}\sum_{i \in I}\sum_{v \in I} w_{iv}^2)}{\partial w_{iv}} + \frac{\partial(\alpha \sum_{i \in I}\sum_{v \in I} w_{iv})}{\partial w_{iv}} \\ &= (r_{uic} - \bar{r}_{uic}) \cdot r_{uv} \cdot sim(c, c_\emptyset) + \beta \cdot w_{iv} + \alpha \end{aligned} \tag{15}$$

Thus, at each iteration of the SGD algorithm, the W parameter will be updated as follow:

$$\begin{aligned} w_{iv} &= w_{iv} - \mu \cdot \frac{\partial Loss}{\partial w_{iv}} \\ &= w_{iv} - \mu \cdot ((r_{uic} - \bar{r}_{uic}) \cdot r_{uv} \cdot sim(c, c_\emptyset) + \beta \cdot w_{iv} + \alpha) \\ &= w_{iv} - \mu \cdot ((r_{uic} - \bar{r}_{uic}) \cdot r_{uv} \cdot (1 - \sqrt{\sum_{k=1}^{D} (c_k - c_{\emptyset k})^2}) + \beta \cdot w_{iv} + \alpha) \end{aligned} \tag{16}$$

w_{iv} is the weight of items i and v. r_{uic} and $\bar{r}_{uic}$ are the predicted and actual rating and r_{uv} is the actual rating of user u toward items i and v. D is the dimensions of context vectors, c and $c_\emptyset$ are context vectors. μ is the learning rate and α and β are regularization hyperparameters that are used for tuning the training process. Context vectors are also parameters that are necessary to learn at each iteration of the SGD algorithm.

$$\begin{aligned} \frac{\partial Loss}{\partial c_k} &= \frac{\partial(\frac{1}{2}\sum_{u \in U}\sum_{i \in I}(r_{uic} - \bar{r}_{uic})^2)}{\partial \bar{r}_{uic}} \cdot \frac{\partial \bar{r}_{uic}}{\partial c_k} + \frac{\partial(\frac{\beta}{2}\sum_{i \in I}\sum_{v \in I} w_{iv}^2)}{\partial c_k} + \frac{\partial(\alpha \sum_{i \in I}\sum_{v \in I} w_{iv})}{\partial c_k} \\ &= (r_{uic} - \bar{r}_{uic}) \cdot (\sum_{v \in I} r_{uv} \cdot w_{iv}) \cdot \frac{c_k - c_{\emptyset k}}{\sqrt{\sum_{k=1}^{D} (c_k - c_{\emptyset k})^2}} \end{aligned} \tag{17}$$

$$\frac{\partial Loss}{\partial c_{\emptyset k}} = \frac{\partial(\frac{1}{2}\sum_{u\in U}\sum_{i\in I}(r_{uic} - \bar{r}_{uic})^2)}{\partial \bar{r}_{uic}} \cdot \frac{\partial \bar{r}_{uic}}{\partial c_{\emptyset k}} + \frac{\partial(\frac{\beta}{2}\sum_{i\in I}\sum_{v\in I} w_{iv}^2)}{\partial c_{\emptyset k}} + \frac{\partial(\alpha\sum_{i\in I}\sum_{v\in I} w_{iv})}{\partial c_{\emptyset k}}$$
$$= -(r_{uic} - \bar{r}_{uic}) \cdot (\sum_{v\in I} r_{uv} \cdot w_{iv}) \cdot \frac{c_k - c_{\emptyset k}}{\sqrt{\sum_{k=1}^{D}(c_k - c_{\emptyset k})^2}} \tag{18}$$

Then the context vectors parameters will be updated as follow:

$$c_k = c_k - \mu \cdot (r_{uic} - \bar{r}_{uic}) \cdot (\sum_{v\in I} r_{uv} \cdot w_{iv}) \cdot \frac{c_k - c_{\emptyset k}}{\sqrt{\sum_{k=1}^{D}(c_k - c_{\emptyset k})^2}} \tag{19}$$

$$c_{\emptyset k} = c_{\emptyset k} + \mu \cdot (r_{uic} - \bar{r}_{uic}) \cdot (\sum_{v\in I} r_{uv} \cdot w_{iv}) \cdot \frac{c_k - c_{\emptyset k}}{\sqrt{\sum_{k=1}^{D}(c_k - c_{\emptyset k})^2}} \tag{20}$$

c_k and $c_{\emptyset k}$ are the values of the k^{th} dimension of context vector c and no-context vector $c_\emptyset$. D is the number of dimensions of context vectors. w_{iv} is the weight of items i and v. r_{uic}, $\bar{r}_{uic}$ are the predicted and actual rating and r_{uv} is the actual rating of user u toward items i and v. μ is the learning rate hyperparameter.

In summary, CSLIM-MCS initializes W (weight) and C (context) parameters annd then finds their optimal values using the Stochastic Gradient Descent algorithm. Then the prediction will be made by the product of the result of the SLIM algorithm and the similarity of context vectors c and $c_\emptyset$.

2.3 Evaluation

2.3.1 Evaluation method

This thesis will use the 5-fold cross-validation method to evaluate the algorithms. In this method, the dataset will be split into 5 parts. Each part will be the validation set at a time and the other parts will be the training set. Then the final result will be the mean of all results.

This thesis also uses several metrics to compare the results:

- *Precision@10* reflects the proportion of true positive results and top 10 results.

  $$prec@10 = \frac{|P_{true}|}{min(10, |P|)}$$

  $|P_{true}|$ is the number of true positive results whereas $|P|$ is the number of results.

- *Recall@10* reflects the proportion of true positive results and the number of results that users selected.

  $$recall@10 = \frac{|P_{true}|}{min(10, |M|)}$$

  $|P_{true}|$ is the number of true positive results, $|M|$ is the numbers of items that users viewed.

- *MAP@10* (Mean Average Precision) is like prec@10 but it can also reflect the correctness of the ranking of results.

  $$MAP@10 = \sum_{1}^{n} \frac{\sum_{i=1}^{10} prec@i}{n \cdot min(10, M)}$$

  n is the number of users, M is the number of items that user views.

2.3.2 Dataset

This thesis will use the InCarMusic dataset [12] to evaluate the algorithm.

Number of Users	Number of Items	Number of Ratings	Rating scale
42	139	3938	1 - 5

Table 1: The information of InCarMusic dataset

The rating scale is from 1 to 5, however, the top-N recommendation problem needs a binary rating matrix as input. Therefore, the rating matrix will be converted to binary values as follow:

$$R^*_{ij} = max(0, min(1, R_{ij} - 2))$$

2.3.3 Evaluation result

To prove the role of contextual information, the SLIM algorithm will be compared with the CSLIM-MCS algorithm. The hyperparameters have been chosen so that the output of each algorithm can be the best:

- *Pre-filtering similarity function*: Pearson correlation
- *Number of neighbors*: 65
- *Number of iterations*: 100
- μ: 0.01
- α, β: 1, 2 for SLIM and 1, 5 for CSLIM-MCS respectively

The below tables and figures show the result of the evaluation using prec@10, recall@10, and MAP@10 metrics.

Top-N	1	2	3	4	5	6	7	8	9	10
SLIM	0.016	0.022	0.026	0.029	0.037	0.041	0.040	0.038	0.038	0.037
CSLIM-MCS	0.047	0.049	0.075	0.066	0.080	0.090	0.083	0.064	0.082	0.086

Table 2: The comparison by prec@10 metric

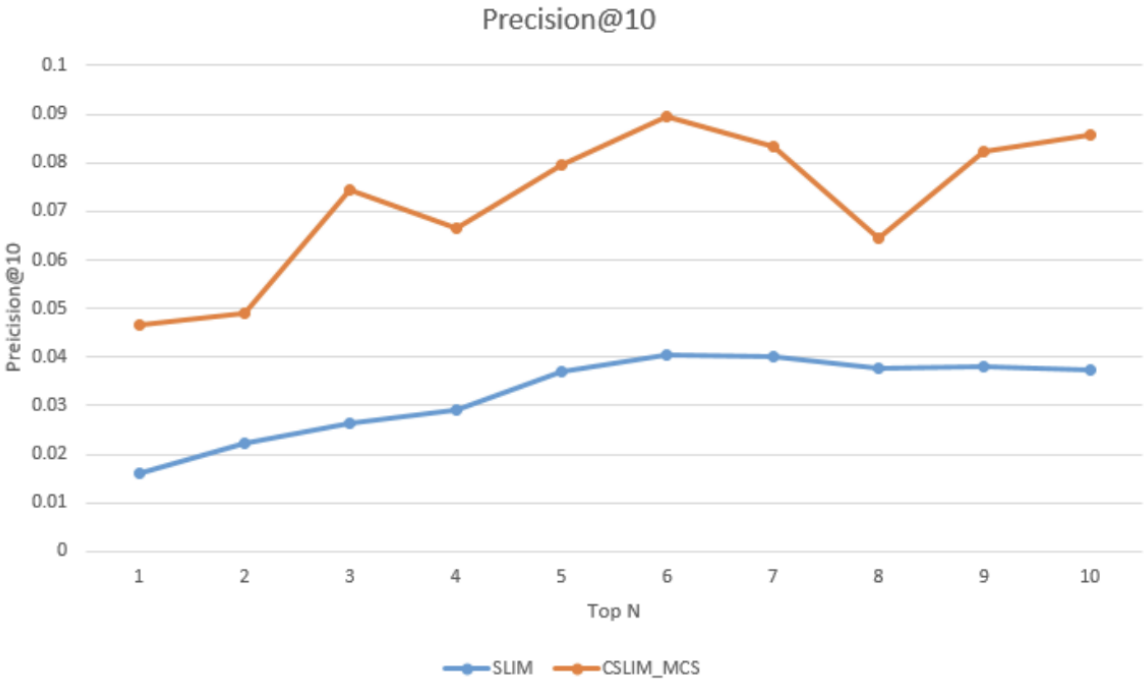

Figure 2: The comparison by prec@10 metric

Top-N	1	2	3	4	5	6	7	8	9	10
SLIM	0.104	0.143	0.166	0.182	0.250	0.258	0.246	0.236	0.232	0.220
CSLIM-MCS	0.245	0.229	0.308	0.302	0.327	0.306	0.360	0.286	0.341	0.403

Table 3: The comparison by recall@10 metric

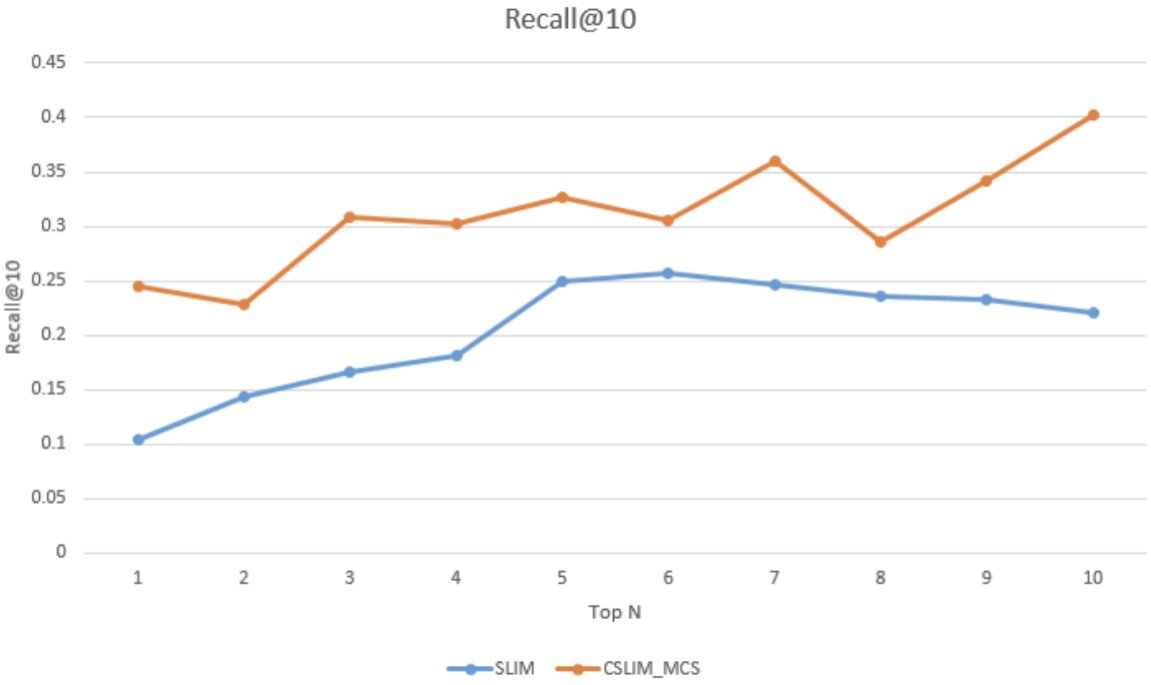

Figure 3: The comparison by recall@10 metric

Top-N	1	2	3	4	5	6	7	8	9	10
SLIM	0.104	0.116	0.124	0.129	0.179	0.161	0.164	0.161	0.163	0.143
CSLIM-MCS	0.245	0.220	0.289	0.277	0.290	0.266	0.331	0.242	0.292	0.312

Table 4: The comparison by MAP@10 metric

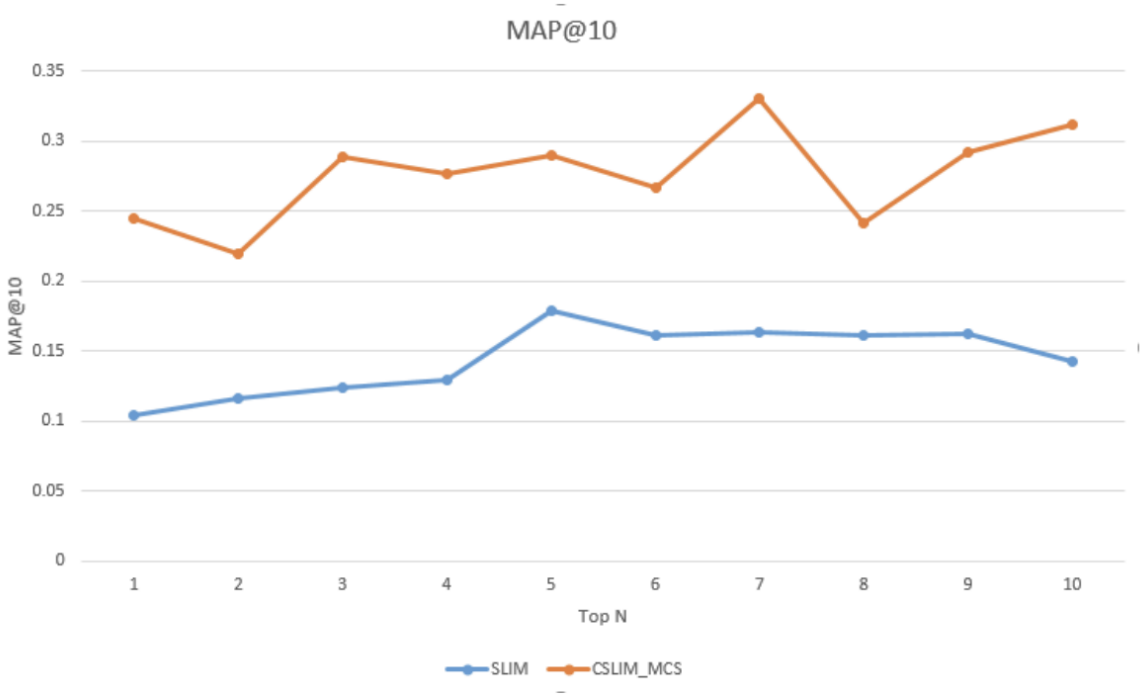

Figure 4: The comparison by map@10 metric

In conclusion, the CSLIM-MCS algorithm has outperformed the SLIM algorithm. This is proof that integrating contextual factors into traditional recommender systems can improve significantly the quality of the recommendations.

References

[1] Francesco Ricci, Bracha Shapira, and Paul B. Kantor. Recommender systems handbook. 2011.

[2] An information-theoretic perspective of tf–idf measures. *National Institute of Informatics*, 2002.

[3] Rummel. Understanding correlation. 1976.

[4] D.Billsus. Learning collaborative information filters. *International Conference on Machine Learning*, 1998.

[5] M.O'Connor. Clustering items for collaborative filtering. 1999.

[6] Xia Ning and George Karypis. Sparse linear methods for top-n recommender systems. 2011.

[7] Gradshteyn and Ryzhik. Tables of integrals, series, and products. 2000.

[8] Bottou and Léon. Stochastic learning. 2004.

[9] TomAlexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. 2010.

[10] Yong Zheng, Bamshad Mobasher, and Robin Burke. Deviation-based contextual slim recommender. 2014.

[11] Yong Zheng, Bamshad Mobasher, and Robin Burke. Similarity-based context-aware recommendation. 2015.

[12] Linas Baltrunas, Marius Kaminskas, Bernd Ludwig, Omar Moling, Francesco Ricci, Aykan Aydin, and Roland Schwaige Karl-Heinz Luke. Incarmusic: Context-aware music recommendations. 2011.