Utilizing Users' Watching Sequences and TV-programs' Metadata for Personalized TV-program Recommendation

DINH QUOC HUNG[†] PAO SRIPRASERTSUK[†] WATARU KAMEYAMA[†]
KENJI FUKUDA[††]

Recently, the explosive growth of digital video contents including IPTV (Internet Protocol Television) has led to the need of recommendation system to guide users among various and huge amount of entertainment movies, live-TV or related services that are called TV programs in general. Consequently, recommendation system has become a general tool to support user's decision in making choice. Most of the ever-proposed algorithms focus on the prediction accuracy; however, we also have to support the diversity of the recommendation results to surprise users in order to widen their choices that might be just missed if the accuracy is only focused on. In this paper, we introduce a new model-based top-K recommendation algorithm called "watch-flow algorithm" for selecting the next *K* highest potential TV programs that user might like. Our model utilizes users' watching sequences and TV program metadata to identify the recommending value for each TV program. Furthermore, this model is also capable of giving a personalized recommendation for a specific user based on his/her watching sequence, as well as capable to improve the prediction accuracy and the diversity. We apply our algorithm on a random sample of users' watching sequences in a dataset collected from real users' log. According to the experimental results, our proposed method shows better performance in recommendation than that of ever-proposed algorithms in terms of higher accuracy while keeping the coverage of programs in high rate.

1. Introduction

With the high-speed development of infrastructure and technology, information systems now are capable of storing and delivering huge information from and to users. It leads to the tremendous growth of information - the concept that formally called "information explosion", which have both merit and demerit. For the merit, with the hosting of thousands or millions of records, all users' preferences are better targeted and satisfied. On the other hand, they also make users confused because of the vast amount of data presented.

IPTV (Internet Protocol Television) is the innovative technology that merges telecommunication and digital television delivery services. Furthermore, IPTV is capable of handling various entertainment movies, live TV or related services such as TV programs in general. Therefore, IPTV is one of the services that both benefited and suffered from information explosion. In order to avoid the downside of information explosion, we need a technique/tool to filter the information or "recommendation system" to guide users to select the most suitable programs among the huge collection of TV programs in IPTV environment.

Over the past decade, recommendation system engine has become discreetly ubiquitous. Until now, there are more innovative works in the area of recommendation system[1] to improve the ability to connect people to the application. In those studies, the accurate and fast interactive recommendation systems based on user historic interactions are exploited deeply. In this paper, we introduce a new model-based top-K recommendation algorithm called "watch-flow algorithm" for selecting the next *K* highest potential TV programs that a user might like. The ultimate goal in our models is not only to improve the prediction's accuracy but also other insights to surprise users. Our proposed model is tested and compared with other recommendation algorithms with the real users' watching history files.

This paper begins with a literature overview of recommendation system in Section 2. We describe our proposed recommendation model and algorithm in Section 3. Subsequently, we give the examples of how the model works in user sequence in Section 4. In Section 5, we introduce our dataset and experimental methodology for our model. Finally, we conclude this paper in Section 6 with a discussion of our future work.

2. Related Works

Generally, recommendation system suggests those most potentially being consumed items to users based on their explicit and implicit preferences. Consequently, recommendation system can be categorized into two different major techniques that are content filtering and collaboration filtering.

The content filtering approach such as [2], [3] and [4] is quite simple, and it requires users to define their own preferences to characterize user's nature preferences. It can be used widely in Web-based services such as Feb system[5] and Syskill and Webert system[6].

In the collaboration filtering approach, the proposal of [7] and [3] are more sophisticated, and there are so many different algorithms used to accomplish this method. According to the study in [8] and [3], memory-based algorithm, vector similarity, and model-based method are effective.

Many works on these research domains are focused on TV program and IPTV recommendation systems. Tsunoda and Hishino[9] employ indirect collaborative filtering and automatic metadata expansion for TV in order to improve the recommendation accuracy. Velusamy et al.[10] applies fuzzy categorical data-clustering techniques to support Ad recommendation system for TV program. Those researches achieve considerable accuracy. However, in Tsunoda and Hishino's model, the parameters of each attribute for each user

[†] GITS Waseda University
[††] WOWOW Inc.

are not easy to determine, and rich metadata needs to be manually manipulated. Moreover, the approach of [10] is focused on Ad context clustering only, thus it cannot take account of user evaluations.

3. Watch-Flow Recommendation System

3.1 Introduction

In this section, we introduce our propose model called watch-flow recommendation model. The proposed model utilizes user historic watching sequences and TV programs' metadata to identify the recommending value for each TV program. The ultimate goal in our model is not only to improve the prediction accuracy but also diversity to surprise users. Thus, it can solve two problems about recommendation system that are recommending performance and novelty or serendipity[11]. To the best of our knowledge, we are the first to attempt to derive a measure of TV program top-K recommending task using real user sequences from large dataset. First, we want introduce all notations used throughout this paper.

$U = \{u_1, u_2 \ \ldots \ u_N\}$: set of N users in which each user holds a sequence of watched TV programs with explicit time length of watching.

$M = \{m_1, m_2 \ldots m_T\}$: set of T TV programs in which each TV program contains metadata information. Metadata of each TV program includes several elements such as genre, 3 main actors, producer, writer, director, and run time.

$G = \{g_1, g_2 \ \ldots \ g_R\}$: set of total R genres that host all TV programs in M. It is noted that each TV program or movie has only one genre in our metadata.

Table 1: Summary of Notations

SYMBOL	DESCRIPTION
Active user	Recommendation list will be computed for this user
K	Top-K recommendations
N	Number of users
u_i	User i in the set of N users
T	Number of TV programs
m_j	TV program j in the set of M TV program
m_j^{genre}	Genre of TV program j
$m_{i,j}^{time}$	Explicit watching time of user i in TV program j
R	Number of genres
g_x	Overall distribution of genre x in dataset (x denotes the genre)
$g_{x,y}^{certain}$	Distribution of genre y after consuming TV program of gen x in dataset
s_i	Number of TV programs in user i sequence
$root_i$	$(s_i - K)^{th}$ TV program in user i sequence. *root* denotes the last TV program user watched before s_i
$similarity_{x,y}$	Overall similarity between TV program x and y calculated by $ubs_{x,y}$ and $mbs_{x,y}$
$ubs_{x,y}$	User-based similarity between TV program x and y
$mbs_{x,y}$	Metadata-based similarity between TV program x and y
r_x	Recommending value for TV program x
p_m	Genre probability of TV program m

3.2 Watch-flow Algorithm

In this Subsection, we describe how to create recommending list of K TV programs for the *active user* based on the recommending value assigned for each TV program.

3.2.1 Recommending Value (r_m)

Unlike the traditional algorithms, the watch-flow takes into account of both user watching sequences and TV programs' metadata. We assume that to the same TV program m, each *active user* u receives different recommending value r_m. r_m is defined as the likelihood estimation that the *active user* will watch TV program m based on the previous one (TV program) called *root*. Moreover, recommending value is associated with the genre probability (p_m) and the similarity between m and *root* $(similarity_{root,m})$.

p_m is the probability that the genre of TV program m can be selected among the set G. In addition, $similarity_{root,m}$ is estimated by the sum of user-based-similarity $(ubs_{root,m})$ and metadata-based-similarity $(mbs_{root,m})$. Consequently, recommending value of TV program m (r_m) is calculated by the following equation (1):

$$r_m = p_m \times (ubs_{root,m} + mbs_{root,m}) \qquad (1)$$

In 3.2.2, we describe in details the calculation of p_m, $ubs_{root,m}$, $mbs_{root,m}$, respectively.

3.2.2 Genre Probability (p_m)

The genre probability (p_m) value reflects the probability of the genre m that likely to happen (selected by the *active user*) after selecting *root* according to the *active user* and other users' genre selection histories. Thus, it equals to the trending value of current genre selection (or current distribution $g_{root,m}^{certain}$) and the total trending value of genre selection (or overall distribution g_m). Therefore, we get equation (2):

$$p_m = g_m + g_{root,m}^{certain} \qquad (2)$$

The overall distribution of genre m (g_m) describes the probability of genre m happened in all user's watching histories. Thus g_m is calculated by the equation (3):

$$g_m = \frac{\#\ Selected\ \ genre\ m}{\#\ Selected\ other\ genres} \qquad (3)$$

In addition, certain genre distribution of genre m after genre *root* $(g_{root,m}^{certain})$ describes the *active user*'s tendency in selecting genre after genre *root* during his/her history. Hence we have the equation (4) for calculating the $g_{root,m}^{certain}$:

$$g_{root,m}^{certain} = \frac{sel_{root,m}}{sel_{root,other}} \quad (4)$$

Where, $sel_{root,m}$ is the number of selected genre *m* after genre *root* in active user history, and $sel_{root,other}$ is the number of selected other genres after genre *root* in active user history.

3.2.3 TV Program Similarity *(Similarity$_{root,m}$)*

In order to discover the inclination in user's watching preferences, we need to learn from the past watching histories to find out those components that user likely to prefer after a particular TV program selection. Therefore, we calculate the similarity between the two TV programs *root* and *m* by the sum of the two distinguish aspects: the user-based *(ubs$_{root,x}$)* and metadata-based *(mbs$_{root,x}$)* similarities. Hence:

$$Similarity_{root,m} = ubs_{root,m} + mbs_{root,m} \quad (5)$$

The user-based similarity *(ubs$_{root,x}$)* value defines the likelihood that users spend much time to watch *m* after *root*. Thus, we have to consider not only the probability of selecting *m* after *root* but also the time length that consumed by *m*. According to the original conditional probability-based similarity[12], in order to calculate the similarity between two items, we extend the formula and apply it to the $ubs_{root,m}$ by considering the TV program play time of *root* and *m*, and user's explicit watching time of TV program *m*. The modified formula is shown in the following equation (6) where α is a parameter that takes a value between 0 and 1.

$$ubs_{root,m} = \frac{sum_{root,m}}{(selected_{root}) \times (selected_{m})^{\alpha}} \quad (6)$$

Where, $sum_{root,m}$ is the total watching time of *m* when user already watched *root*, $selected_{root}$ = number of selected *root* × play time of *root* and $selected_{m}$ = number of selected *m* × play time of *m*.

Additionally, the metadata-based similarity *(mbs$_{root,m}$)* value reflects the similarity in details of *root* and *m*. This value is utilized in order to find out the tendency of the next TV program's metadata after *root* by considering the metadata elements in both *root* and *m*. Therefore, the $mbs_{root,m}$ is calculated by (7):

$$mbs_{root,m} = \frac{intersection_{root,m}}{\#\ metadata\ elements} \quad (7)$$

Where, $intersection_{root,m}$ is the number of intersection of *root* and *m* metadata elements.

3.2.4 Creating Top-K Recommendation List

Based on the above parameters and formulas, we can create a recommending list of *M* recommending value *r* for every single user by using equation (1). After sorting the recommending list descending by *r*, we can get a top-K recommendation list by picking up the *K*-top ranked TV programs from the list.

4. Example of Using Watch-Flow Algorithm

4.1 Example of Using Scenario of Watch-flow Algorithm

User$_1$ watching sequence, which is taken as the ground truth, with s_1 (seven) TV programs/movies, is shown in the format of *Title (time, genre, actor1, actor2)* as follows:

1. Mr. & Mrs. Smith (60, action, Pitt, Jolie),
2. Troy (50, action, Pitt, Bana),
3. Harry Potter (45, adventure, Daniel, Emma),
4. Titanic (40, romantic Leo, Kate),
5. Salt (25, action, Jolie, Elve),
6. Moneyball (30, romantic, Pitt, Wright),
7. Avatar (50, action, Sam, Joe)

The above data can be represented by the two sets of data as follows:

- M = {Mr. & Mrs. Smith, Troy, Harry Potter, Titanic, Salt, Moneyball, Avatar}
- G = {action, adventure, romantic}

From *user$_1$* with s_1 TV programs in a sequence, we split it into the following two smaller sequences:

- The first one is training sequence,
- The second one is test sequence containing K TV programs.

And the last TV program in the training sequence is called *root* for *user$_1$*. Thus, we have:

- $s_1 = 7$
- K = 3
- Root = Titanic

Therefore, we have the model for training and testing with one user in the watch-flow algorithm as shown in Figure 1.

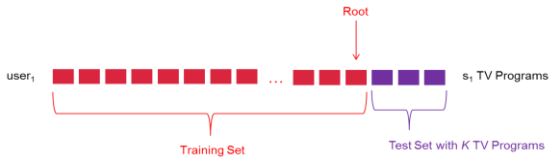

Figure 1. Watch-Flow Scenario for One User

Moreover, we now suppose that:

- We have *N* users: *user$_1$*, *user$_2$*, …, *user$_N$*
- Dataset *M* has *T* TV programs, and,
- Dataset of genre *G* has *R* genres.

A training set is composed of *N* training sequences, and a test set is composed of *N* test sequences. We finally have the watch-flow scenario for *N* user as shown in Figure 2.

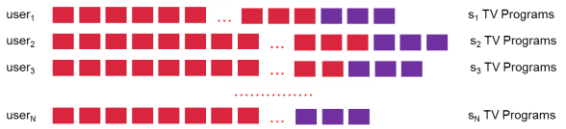

Figure 2. Watch-Flow Scenario for *N* User

From the training set, we now try to predict the test sequence for each user in the corresponding test set by conducting the watch-flow algorithm mainly by the two steps as follows:

1. Compute personalized recommending list: combine all the *M* TV programs with user's personal activities to create a list of *M* recommending values *r*.
2. Then, make recommending list descending by *r*. We can get a top *K* recommendation list by picking the *K* top ranked TV programs from the list.

4.2 Example of Using Watch-Flow Algorithm and Result

Suppose that we have 10 TV programs in dataset together with recommending value gained by the watch-flow algorithm: $\{(m_1, r_1), (m_2, r_2), (m_3, r_3), (m_4, r_4), (m_5, r_5), (m_6, r_6), (m_7, r_7), (m_8, r_8), (m_9, r_9), (m_{10}, r_{10})\}$.

Then, by sorting the list by recommending value, we have: $\{(m_{10}, r_{10}), (m_8, r_8), (m_4, r_4), (m_5, r_5), (m_1, r_1), (m_2, r_2), (m_7, r_7), (m_3, r_3), (m_6, r_6), (m_9, r_9)\}$

Among them, supposing that the *active user u* already watched: $\{m_2, m_4\}$. Now, by selecting top 3 to recommend to the *active user u*, the final recommending list would be obtained by removing m_4 and m_2. Therefore, the top 3 recommendations are: $\{m_{10}, m_8, m_5\}$.

5. Evaluation of the Algorithm

5.1 Experimental Dataset

We evaluate the watch-flow algorithm using the data of user access to the WOWOW Website, where TV-program or movie information is available. We take the specific Webpage watching featuring specific movie information by a user as a real TV program or movie watching by him/her in our evaluation. The dataset stores all the user activities, where each user can be identified by a unique cookie without knowing any information of his/her privacy. It includes the details of all the information that need to evaluate our model, such as the starting time of a Webpage browsing, the time duration for watching each Webpage, and also metadata for every TV program in the dataset. Using the cookie, we can easily and properly distinguish each user watching sequence for a long time. The following items are some statistical properties for this dataset:

- Number of TV programs: 10132
- Metadata elements of each TV program: actors, director, production, writer, …
- Total genres: 50
- Each moth: averagely 1M interaction records with more than 200,000 active users.
- The duration for the dataset is 2 years from September 1st 2009 to September 30th 2011.

5.2 Testing Methodology

We first process the dataset by removing all TV programs that watched less than 30 seconds to avoid zapping. If a user watches them more than or equal to 30 seconds, it ensures that the user see them in detail with very interest.

From the original dataset, we randomly choose 10 data groups in which each group contains 7 days data and more than 2000 users for the experiment. In addition, each user in the data group must have more than 10 TV programs in the watching sequence. The final experiment result is to be obtained by the average result of these 10 data groups.

We test our proposed algorithm in top-K recommending list with the *K* runs varying from 1 to 10. The illustration of this methodology is show in Figure 3.

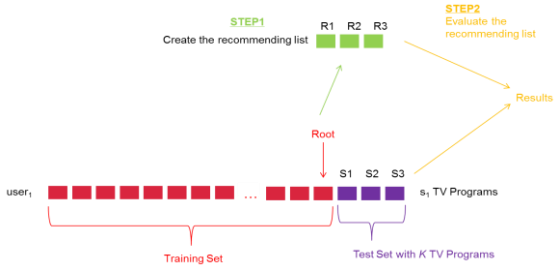

Figure 3. Watch-Flow Experimental Methodology

5.3 Recommendation Evaluation Method

5.3.1 Evaluation Metrics

We use 4 metrics to evaluate how effective the watch-flow algorithm is in each single active user. The *selecting rate* (equation (8)) measures the precision in selecting titles in the *K* recommended titles. The selecting *time rate* (equation (9)) measures the fraction between the total watching times of titles precisely chosen by user $(totaltime_{hits})$ and the total watching time of *K* titles in the recommending list $(totaltime_k)$. The selecting *genre rate* (equation (10)) measures the accuracy in the recommended genre in the recommending list. Finally, the *coverage increasing* (equation (11)) measures the fraction between the frequency of new features (*fre)* and the frequency of old features $(frequency_{user})$. *fre* reflect the relative complement of features in the recommending list in user's watching historic features. The feature in the coverage is decided by the metadata components. The *coverage increasing* is for judging how novel each recommendation can be recommended by the algorithm.

$$Selecting\ Rate = \frac{\#\ of\ hits\ in\ title}{K} \quad (8)$$

$$Time\ Rate = \frac{totaltime_{hits}}{totaltime_k} \quad (9)$$

$$Genre\ Rate = \frac{\#\ of\ hits\ in\ genre}{K} \quad (10)$$

$$Coverage\ increasing = \frac{fre}{frequency_{user}} \quad (11)$$

5.3.2 Evaluation Example

Regarding to the example of using the watch-flow algorithm in 4.1, we explain the evaluation process. Suppose that we have the recommending list with *K* = 3 for the active user in 4.1 created by the watch-flow in the format of *Title (playtime, genre, actor1, actor2)* as follows:

Spiderman (action, 60, Tobey, Kristen),
Salt (action, 30, Jolie, Elve),
Moneyball (romantic, 40, Pitt, Wright)

Based on our definition, in the *active user* sequence, TV programs from *Mr. & Mrs. Smith* to *Titanic* are for the training set, thus *Titanic* becomes the *root*, and the test set is from *Salt* to *Avatar*.

Now, applying equation (8), (9), (10) and (11) for evaluation, we have the final evaluation results as follows:

$$Selecting\ Rate = \frac{\text{Salt} + \text{Moneyball}}{\text{Spiderman} + \text{Salt} + \text{Moneyball}} = \frac{2}{3}$$

$$Time\ Rate = \frac{\text{Salt} * 25 + \text{Moneyball} * 30}{\text{Spiderman} * 60 + \text{Salt} * 30 + \text{Moneyball} * 40} = \frac{11}{26}$$

$$Genre\ Rate = \frac{\text{action} + \text{romatic} + \text{action}}{\text{action} + \text{action} + \text{romantic}} = \frac{3}{3}$$

$$Coverage\ Increasing = \frac{(\text{Tobey}*1+\text{Kristen}*1+\text{Elve}*1+\text{Wright}*1)/4}{(\text{Pitt}*2+\text{Jolie}*1+\text{Bana}*1+\text{Daniel}*1+\text{Emma}*1+\text{Leo}*1+\text{Kate}*1)/7} = \frac{1}{1.14}$$

5.4 Competitors' Base Lines

In order to evaluate our algorithmic, we consider the several common and the state-of-the-art recommendation algorithms for comparison[13, 14] as following:

- Non-personalized algorithm or top popular *(TopPop)*: *TopPop* implements a simple estimation algorithm. It chooses the most watched TV programs to recommend to any user, regardless his/her historic usage.
- Non-personalized algorithm or movie average *(MovieAvg)*: *MovieAvg* recommends the top-K items with the highest average of watching time.
- Neighborhood model or non-normalized cosine neighborhood model *(NNCosNgbr)*: *NNCosNgbr* computes TV program similarity by means of the coefficient. Unlike Pearson correlation which is computed only on ratings shared by the common rater, the cosine coefficient is computed over all ratings, and it takes missing values as zeroes.
- Latent factor model: based on previous study on top-K recommending, the recently proposed *PureSVD150* which is based on the conventional Singular Value Decomposition (SVD) with 50 factors is a latent factor algorithm that yields the highest accuracy compared to other advanced latent factor model.
- Conditional probability-based similarity *(CPbS)*: the equation (6) to calculate the user-based-similarity is inspired by this *CPbS* equation. Thus, it is also a base line for us to compare this algorithm. This base line uses $\alpha = 0.5$ as stated in the original paper[12].

5.5 Experimental Results

In this Subsection, we report experimental results comparing the watch-flow and others algorithms. From the original dataset, after the process as described in 5.2, there are more than 2000 users per period left for the testing. Those results described below are taken by averaging all results from every single *active user*. For other competitors' algorithms, if the algorithm requires the rating as input for a specific TV program, we simulate the rating by the fraction between the *active user*'s explicit watching time length and the average of all users' watching time length of that TV program, and scale it up to rating from 1 to 5 as in the original works, because our dataset do not have the rating. Furthermore, in recommending TV program to user, $K = 10$ is enough, thus we only test running with K from 1 to 10.

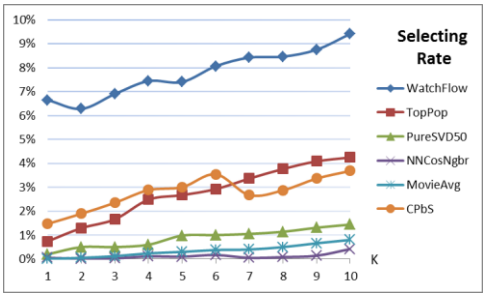

Figure 4. Comparison Results for Selecting Rate

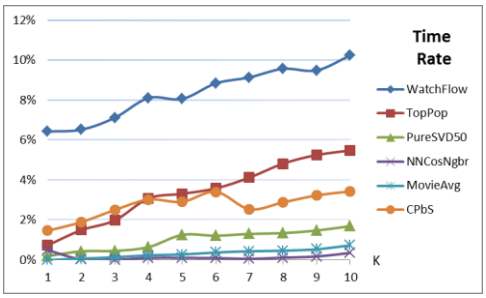

Figure 5. Comparison Results for Time Rate

For the *selecting rate* and *time rate* evaluations as shown in Figure 4 and 5, it is clear that the watch-flow algorithm absolutely outperforms the others. When increasing K, the watch-flow algorithm improves the accuracy. There are also tendency of increasing the accuracy observed in other algorithms. However, they are much lower compared with the watch-flow algorithm. For instance, at $K = 10$, within other 5 algorithms, *TopPop* shows the highest selecting rate. Surprisingly, the watching-flow algorithm is about twice better than that of *TopPop*. Also, we can see that in the time rate comparison, the accuracy of the watch-flow algorithm always increases together with the selecting rate, and the time rate is almost higher than selecting rate in the same K. That makes the shape of time rate is similar to the shape of selecting rate as expected.

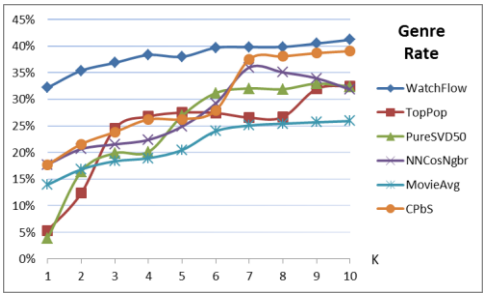

Figure 6. Comparison Results for Genre Rate

For the *genre rate* evaluation as shown in Figure 6, the more *K* means the more accurate that all algorithms can achieve for selecting the precise genre to be recommended. The watch-flow algorithm yields better results than others. From *K* = 7 to 10, the watch-flow algorithm does not increase the accuracy as fast as smaller *K*s do, and *CbPbS* can get the accuracy around 3% lower than the watch-flow algorithm from *K* = 7 to 10.

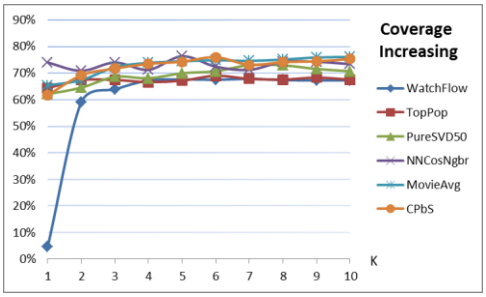

Figure 7. Comparison Results for Coverage Increasing

Finally, for the *coverage increasing* evaluation as shown in Figure 7, the more *K* again shows the more coverage of the recommending list. We can observe from Figure 7 that the increasing in coverage potentially raises the trade-off in precision. In fact, *MovieAvg* and *NNCosNgbr* give the lowest precision in both *selecting rate* and *time rate*, and they give the highest results in *coverage increasing*. Therefore, the watch-flow algorithm has the lowest result in *coverage increasing* together with *TopPop*, even though the watch-flow algorithm gives the best accuracy in *selecting rate* but *TopPop* does not. Furthermore, the *coverage increasing* in the watch-flow algorithm is acceptable because the difference between the highest one which is *MovieAvg* and the watch-flow algorithm is only around 8% except when *K* = 1. Therefore, we consider that the watch-flow algorithm can balance between fitting user's previous preferences and recommending new features or novelty. And, it is capable of surprising user with unexpected recommendations in the least of trade-off with accuracy in both *selecting rate* and *time rate*.

Comparing the performance of other 5 algorithms on a top-K recommendation task, our results show that the proposed algorithm improves the quality of recommendation including accuracy and diversity. It also proves again that the accuracy in predicting rating does not relevant with the accuracy in top-K selection in a practical use.

6. Conclusion and Future Work

In this paper, we propose the watch-flow algorithm recommending TV programs. Our algorithm takes advantage of user watching sequences to decide TV programs that are more likely to be selected by users to watch with the highest the time spend.

For the future improvement of this algorithm, we will investigate those TV program with watching-time less than the thresh-hold of 30 seconds. In the current experiment, we remove those TV programs in the user sequence and consider them as user's dissatisfaction. Hence, it is necessary to study what factors that dissatisfy a user and provide better recommendation for them. Moreover, the current results are within 7 days duration, we need to test with the longer time duration in order to see the discrepancy of both accuracy performance and diversity performance. Furthermore, we will analyze the change of performance in each time-frame during a day. Finally, we will also investigate and analyze the user behavior and the emotion during watching TV programs to find a usage pattern to realize social IPTV system.

Reference

1) Bennet, J. and Lanning, S.: The Netflix Prize, KDD Cup and Workshop (2007).
2) Zhang, T.: Solving Large Scale Linear Prediction Problems Using Stochastic Gradient Descent Algorithms, in ICML 04, pp.919–926, (2004).
3) Adomavicius, G. and Tuzhilin, A.: Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions, IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 6, pp.634-749, (2005).
4) Belkin, N. and Croft, B.: Information Filtering and Information Retrieval, Comm. ACM, Vol. 35, No. 12, pp.29-37, (1992).
5) Balabanovic, M. and Shoham, Y.: Fab: Content-Based, Collaborative Recommendation, Comm. ACM, Vol. 40, No. 3, pp.66-72, (1997).
6) Pazzani, M. and Billsus, D.: Learning and Revising User Profiles: The Identification of Interesting Web Sites, Machine Learning, Vol. 27, pp.313-331, (1997).
7) Gunawardana, A. and Shani, G.: A Survey of Accuracy Evaluation Metrics of Recommendation Tasks, in Journal of Machine Learning Research, Vol. 10, pp.2935-2962, (2009).
8) Breese, J. S., Heckerman, D. and Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering, in Proc. of the 14th Conference on Uncertainty in Artificial Intelligence, pp.43–52, (1999).
9) Tsunoda, T. and Hoshino, M.: Automatic Metadata Expansion and Indirect Collaborative Filtering for TV Program Recommendation System, Multimedia Tools and Applications, Vol. 36, pp.37-54, (2008).
10) Sarwar, B., Karypis, M., Konstan, J. and Riedl, J.: Item-Based Collaborative Filtering Recommendation Algorithms, in Proc. of the 10th Int'l World Wide Web Conference (WWW 01), ACM, pp.285–295, (2001).
11) Herlocker, J.L., Konstan, J.A., Terveen, L.G. and Riedl, J.T.: Evaluating Collaborative Filtering Recommender Systems, ACM Trans. Information Systems, Vol. 22, No. 1, pp.5-53, (2004).
12) Karypis, G.: Evaluation of Item-Based Top-n Recommendation Algorithms, in Proc. Of the 10th International Conference on Information and Knowledge Management (CIKM '01), pp.247–254, New York, NY, USA, (2001).
13) Cremonesi, P., Koren, Y. and Turrin, Y.: Performance of Recommender Algorithms on Top-N Recommendation Tasks, in Proc. of the 4th ACM Conference on Recommender Systems (RecSys '10), pp.39–46, ACM, Barcelona, (2010).
14) Cremonesi, P., Garzotto, F., Negro, S., Papadopoulos, A. V. and Turrin, R.: Looking for "Good" Recommendations: A Comparative Evaluation of Recommender Systems, INTERACT 2011, Lecture notes in computer science, Vol. 6948/2011, pp.152-168, (2011).