Recommendation System based on Predictive Approach

Ibrahim Hussein Mwinyi, Husnu S. Narman, Kuo-Chi Fang, Wook-Sung Yoo

TR-MU-CITE-17-100
Oct 2017

Department of Computer Science

College of Information Technology and Engineering

MARSHALL UNIVERSITY

Arthur Weisberg Family Applied Engineering Complex 3107, One John Marshall Drive, Huntington, WV 25755
(304)-696-5829, narman@marshall.edu, http://hsnarman.oucreate.com/

Recommendation System based on Predictive Approach

Ibrahim Hussein Mwinyi, Husnu S. Narman, Kuo-Chi Fang, and Wook-Sung Yoo
Computer Science, Marshall University, Huntington, WV 25755
Email: {mwinyi, narman, fang5, yoow}@marshall.edu

***Abstract*—Millions of users use the Internet for entertainment, education, shopping and many other purposes. For instance; one billion hours of YouTube videos are watched every day. One of the key features of such platforms such as the entertainment and shopping platforms is the recommendation system based on past activities of users and the contents of the visited sites to provide related contents to decrease search time and increase the data availability. However, the content suggestions have several challenges based on the platforms. Some challenges are: collected data from users can be noisy, the media contents are not well-defined, and users do not want to cooperate. To solve those issues, YouTube recommendation, Netflix, AWS re:invent, and similar commercial sites proposed the context-aware personalized recommendation systems. Most of the recommendation systems use implicit and explicit user past activities with mapping relation between contents. However, the recommendation systems are general and cannot be changed according to user characteristics after visiting the suggested data. For example, a user mostly visits the higher-ranking content which is related to the visited content, and another user can visit the recent content which has high feedback during the different days or even in an hour according to mood. Therefore, in this paper, we propose a predictive self-learning recommendation system. The algorithm predicts what a user searches next by using prior collected information and using machine learning to analyze the user behaviors for the future activity. The results show that our proposed recommendation system is efficient in terms of CPU usage and response time while characterizing users' behaviors in short and long terms. The proposed method and related analysis can assist the shopping, entertainment and similar recommendation systems to increase their efficiency by well-characterizing users' behaviors.**

***Index Terms*—Prediction, recommendation, social media, profiling**

I. Introduction

A large number of users use the Internet for entertainment, education, shopping and many other purposes. For example; one billion hours of YouTube videos are watched per day [1]. One of the key features of such platforms such as the entertainment and shopping platforms is the recommendation system based on past activities of users and the contents of the visited sites to provide related contents to decrease search time and increase the data availability. However, the content suggestions have several challenges based on the platforms.Some challenges are: collected data from users can be noisy, the media contents are not well-defined, and users do not want to cooperate [2]–[11].

To solve those issues, YouTube recommendation, proposed a personalized recommendation system and grew to become the largest online video community with vast amounts of videos being watched daily [12]. Authors [12] present a unique problem which our proposed algorithm attempts to solve [12]. Some of those challenges include the fact that videos are uploaded by individual users, thus have little or no meta data (which is required for recommendation systems). In addition, uploaded videos can be distributed to many users in a matter of seconds which requires constant updates to the recommendations [12]. Netflix uses a combination of recommendation algorithms to create the "Netflix Experience", a subscription-based system that has over 65 million users streaming 100 million hours of movies daily [13]. AWS re:invent is Amazon's recommendation system made up of various recommendation algorithms in order to personalize the online store for each individual user [14]. The Amazon online store drastically changes based on customer interests. The main problems that Amazon faced and solved with the implementation of their recommendation system include vast amounts of data including millions of customer information, billions of product data that all have to be compressed into meaningful recommendations presented to the user in real time [14].

The recommendation systems listed above are general and cannot be changed according to user characteristics after visiting the suggested data. YouTube recommendation system considers several parameters such as the related content, popularity, channels, location, past activities, language, user profiles, and time period while recommending videos to users [12]. Netflix uses not only local media related videos but also global related videos while suggesting to users because Netflix subscribers spend only 60 to 90 seconds to find a video to watch or the subscribers move another activity such as reading book [13]. Amazon recommendation system suggests the similar type object as well as the object collection which was bought by other users in the recommendation system [14]. However, the similar approaches identify the long-term behaviors of the users but limit to identify the short-term behaviors. For example, a user mostly visits higher-ranking contents out of suggested contents today and the newest related contents out of suggested contents tomorrow. Another user can visit the recent content which has high feedback during the different days or even in an hour according to the mood. Therefore, the *aim* of this paper is to propose a predictive self-learning recommendation system that takes into account users' preferences before and after selections are made. Recommendations are predicted by the system ahead of time in order to quickly present suggestions to users when they need without diminishing the meaningfulness of the suggestions. The system gradually learns the user's personal preferences through their selection habits and eventually narrow down those preferences to a set of categories. To maintain the diversity of the suggested contents, the similar approach with [12], [13] has been followed.

The *objective* of this paper is to create an algorithm that predicts what users search next by using prior collected information and using machine learning to analyze the user behaviors for the future activity. Then, we apply a behavior analyzer to update the prediction system by monitoring users selections from suggested contents.

We use seven parameters which must be satisfied to make a successful prediction. The parameters include; popularity [15], similarity [15], currency [15], feedback [12], importance [12], interest [13], and safeness [12] in addition to user profiles [7]. Popularity refers to data that is currently trending in this location and worldwide [15]. Similarity is where the machine learning algorithm is used as the system needs to identify similar data within the data centers and return the most related ones and increases the chances of a correct prediction [15]. Currency means how old or new the content is, therefore returning the contents from a suitable period as needed by the user [15]. Feedback helps the system establishes the importance and safeness allowing users to acquire content that is both important and safe [12]. Interest in data varies from user to user, for example, a doctor is interested in medical contents while a musician is interested in music [13]. Using some or all of these criteria, the system searches our servers for appropriate contents by using a tag system and returns what are needed for users. Then, we apply the user behavior analyzer to update the prediction system to improve the quality of the suggested contents. The results show that our proposed recommendation system is efficient in terms of CPU usage and response time while characterizing users' behaviors in short and long terms.

The key *contributions* of this paper can be listed as follows:

- A predictive self-learning recommendation system which improves the suggested content awareness for each individual has been proposed.
- An extensive simulation has been developed by using real-data with generated required parameters to analyze the performance of the proposed recommendation system.
- The proposed system has been analyzed in terms of the system performance, the system efficiency, and the system integrity.

The rest of the paper is organized as follows: Section II describes a general model of recommendation system. In Section III, we explain the used parameters for the proposed system. Section IV describes the used technique while suggesting contents. Section V explains the methodology to test our proposed recommendation system and Section V consists of the results. Finally, Section VII includes the final discussion and our plan to extend this work.

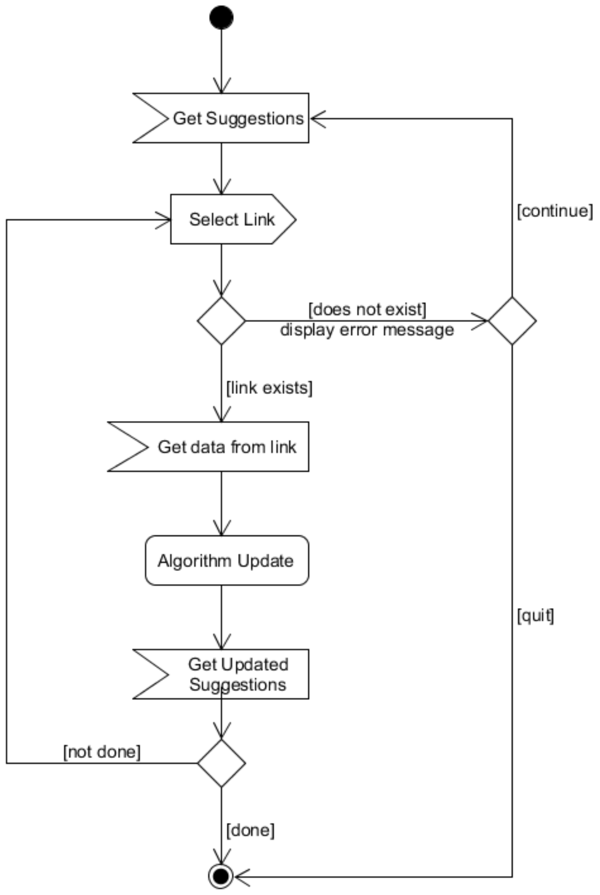

Fig. 1. A user activity when interacting with the system.

II. System Model

Fig. 1 shows a user interaction with the suggested contents. After suggestions are made, the user visits one of the suggested contents. It is possible that the selected content does not exist anymore because of consistency issues. If the content exists, the recommendations are updated.

The predictive self-learning recommendation system is based on a predictive algorithm that would greatly decrease a user option selection complexity. The development of this algorithm required a number of factors such as; recognizing the algorithm criteria, research on content suggesting parameters, and creating a measurement system for parameters and the recommendation. The algorithm is based on the following criteria; popularity, similarity, currency, feedback, importance, interest and safeness in addition to user profiles. Based on those criteria, a successful prediction can be made [12].

Popularity of certain data is calculated by using trend analysis and overall clicks. Trend analysis is when the system analyzes the top trending data on the web and record that data for future use. Overall clicks refer to the number of times that data has been accessed. Similarity is acquired through the tag system within the database. The database holds all available data and labels each set of data with one or multiple tags. Therefore, the similar data have the same tags or the same category tags. Another way to acquire similarity between the data is with the use of a machine learning algorithm. The collaborative filtering algorithms are used on most recommendation systems [3], [13] in recent years. Currency refers to the data age and depends on how users utilize the data; currency could be a big factor in making a prediction. Feedback is what users provide the system once the data has been accessed. Users' feedbacks are completely optional and can be a ”LIKE” or ”UNLIKE” or ranking formats. Importance and safeness of the data also are established through user feedback. Users can provide additional details on a particular set of data; those additional details include; safeness rating of the data (how safe the data is) and importance rating of the data (how important the data is). Finally, interest is acquired using a combination of some criteria we have discussed above. The interest rating is the unique interest that users have on a set of data. Therefore, users interest in a set of data can be calculated through the combination of popularity, currency, feedback, importance, and safety as well as factoring in background information given to the system initially by the user. Once all ratings have been acquired, an overall rating for the user on a specific set of data is calculated. Calculation of all ratings is discussed in the next section.

III. System Metrics

The overall ranking of a set of data which is used in this paper:

$$R = \alpha Uc + \beta(Fp - Fn) + (\gamma \frac{Pl}{10} + \eta \frac{Po}{10} + \kappa \frac{Pn}{10} + \varsigma \frac{Pi}{10}) + \psi Oc \quad (1)$$

where $\alpha, \beta, \gamma, \eta, \kappa, \varsigma$ and ψ are constant and:

- R (Link Ranking) - The overall ranking of the content that the user has clicked. The overall ranking changes from user to user in order to provide a more personalized experience for each individual user.
- Uc (User Clicks) - The number of times the user has clicked the link. This is considered to learn about the user's specific preferences. As this number increases, the overall ranking of the link (for the specific user) also increases, therefore more likely to be suggested.
- Fp (Positive Feedback) - The number of times the content has received positive feedback from all users in addition to the user past feedbacks for the same type of contents. This is taken into account because the system needs to learn the general opinion about users' opinions to decide whether to suggest it or not. Positive feedback increases the suggestion rate.
- Fn (Negative Feedback) - The number of times the content has received negative feedback from all users in addition to the user past feedback for the same type of contents. Similar to positive feedback, negative feedback is considered because the system needs to learn the general opinion about users' opinions to decide whether to suggest it or not. Negative feedback decreases the suggestion rate. Therefore, (Fp-Fn) is the overall feedback (general opinion on importance and safety of the link) by all users. If it is negative, data has been more disliked than liked, therefore less likely to be suggested by the system.
- Pl (Location) - The user can provide us with some background information prior to using the system. This background information includes the location of the user which is considered by the system when returning suggestions. It is compared to the upload location of the content and given a score out of 10. For example; location score can be 5 out of 10 if a user lives in the United States, but the content is originally uploaded from England. The closer the proximity, the higher score.
- Po (Occupation) - Similar to location, the occupation of the user which is considered by the system when returning suggestions. The category of the content is compared and given a score out of 10.
- Pn (Nationality) - The language of the content and the access location are considered and given a score out of 10.
- Pi (Interests) - The user can select several interests before the suggestion made. Therefore, the interests of the user is a factor while returning suggestions and given a score out of 10.
- Oc (Overall Clicks) - It refers to the number of times all users have accessed the content, gives the system a general picture on the popularity. The higher number of accesses (trending), the higher overall ranking, therefore, the more likely the content is recommended.

The metrics above are used as a base for building a prediction tailored to each individual user as well as to verify the integrity of individual prediction. The algorithm engine makes use of these metrics in order to make a customized prediction for each individual user. The engine is discussed in the following section.

Moreover, equation (2) shows the Collaborative Filtering Machine Learning Algorithm [16] which is used to improve the prediction algorithm over time. Given $\theta^{(1)}, ..., \theta^{(n_u)}$ to learn $x^{(1)}, ..., x^{(n_m)}$,

$$\begin{aligned} \min_{x^{(1)},...,x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 \\ + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 \end{aligned} \quad (2)$$

Equation (2) illustrates that given a matrix of user movie genre preferences and a matrix of movie genre's, one can deduct the type of movie genre according to the user preferences and vice versa [16]. Therefore, with the continued deduction of both, the algorithm constantly updates itself and objects in the recommendation system allowing more accurate categorization of the objects. We used the similar approach

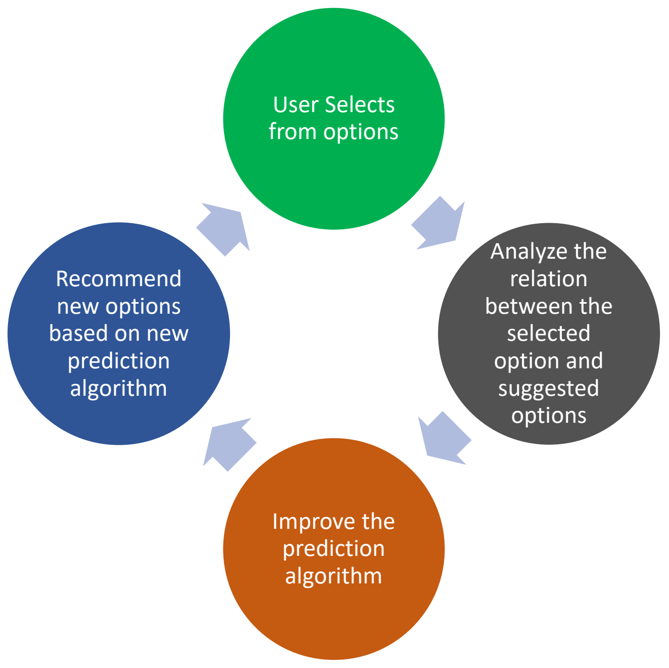
User Selects from options
Analyze the relation between the selected option and suggested options
Improve the prediction algorithm
Recommend new options based on new prediction algorithm

Fig. 2. Recommendation improvement model for content suggestions.

while improving our recommendation when a user selects the suggested contents.

IV. System Engine

As mentioned in previous sections, the recommendation requires several criteria such as popularity, similarity, currency, feedback, importance, interest, and safety before the system can make suggestions to users. After users select from the suggested contents, we improve the recommendation system by predicting the user short-term behaviors. As shown in Fig. 2, a user interaction flow in the system, a user searches contents or selects from suggestions. Once the user has input for the selection, the system suggests the related contents to the user based on the current information about users from the system database and the above criteria. After the suggestions are completed, and the user selects one option, the system analyzes the relationship between previously suggested options and the current selection of the users with the criteria which are used before to improve the prediction algorithm. Finally, the system recommends new and improved options based on the improved the recommendations.

Algorithm 1 is the key factor to analyze the user short-term behaviors by creating a relation between the previous suggestions and the current selection. While creating a relation, indeed, the double filtering is happening because the same described criteria are used in the suggested contents. However, this time content size is relatively small. Therefore, the better relation is created in almost instant time, and then related contents from all system are filtered for suggestions.

Algorithm 1 Algorithm to improve recommendation systems

```
Require:
        input - user profiles and past activities
        output - suggestions
 1: procedure
 2:     while user not done do
 3:         display suggestions
 4:         get user selections
 5:         if selection in DB then
 6:             display the content
 7:             create a relation: selected and suggested
    options
 8:             update user recommendation system
 9:             display some suggestions according to the
    new recommendation and some based on the previous
    recommendation system
10:         end if
11:     end while
12: end procedure
```

V. Analysis

In this section, the simulation environment, used data, and test cases which have been used to analyze the performance of the proposed system, are explained.

A. Used Data in the Analysis

To test the efficiency of the recommendation system, the real metadata and their links are downloaded from YouTube [17]. The downloaded metadata size is around 10GB. However, the data which downloaded from YouTube does not include the user information because of privacy issues. Therefore, the user type information is generated from the simulation, and these include user profiles, queries, interest, and user own feedbacks for some contents.

B. Test Cases

Analysis of the recommendation system is tested based on three main concepts; system performance, system efficiency, and system integrity. System performance is measured through response time, i.e., how long it takes from when the user makes a selection to when the database is updated, and the result is returned to the user. System efficiency is measured by constantly monitoring CPU Utilization ensuring the system is not be overwhelmed. System integrity is measured by the uniqueness of suggestions that is returned to the user.

C. Simulation Design

In order to test the performance of the recommendation system, we conduct an extensive simulation. The simulation is implemented using Java programming language (Eclipse IDE). Because of data size, one computer with quad-core 2.4 GHz with 16GB memory and 32MB cache is used. In the simulation, we assume that users have already searched for initial suggestions because that part is out of our scope. On the other hand, users' interactions with suggested contents are critical for the simulation. Therefore, we focus on this part during the simulation. The simulation generates a number of users and allow the suggestions to be displayed based on users' selections and past activities which are initially generates, but then updates users' profiles according to their selections. While a user can continuously select one to ten links out of 1000 related contents available in the database, the simulation allows the database to be updated according to selections, allows the collaborative filtering algorithm to be updated, and finally returns updated suggestions to users. In the simulation, simulated users get the current suggestions from the system, ten times select contents, and receive updated suggestions from the system based on the selected contents.

VI. Results

The results which are displayed in this section are the average of 200 simulated realizations. However, while we test the CPU utilization, we only consider the first realization because the remaining realizations decrease the CPU Utilization due the operating system optimization as shown in Fig. 3. To test the CPU utilization, the first realization was performed with an overall execution time of 60s to observe the overhead of the algorithm. As presented in Fig. 3,

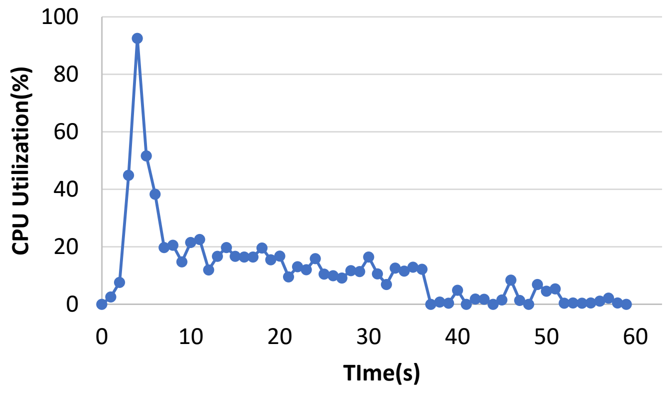

Fig. 3. CPU Utilization while running the recommendation system.

CPU utilization initially very high because the simulation is started, but then the utilization slowly decreases as expected. We observe that in 35s, hundred simulated users get the current suggestions from the system, orderly select ten contents, and receive updated suggestions from the system based on the selected contents. Because of 35s operations, CPU utilization decreases after that time. From Fig. 3, we observe that the CPU utilization is changing between 10% to 20% for this simulation. From this data, we can conclude that the predictive self-learning recommendation system does not demand too much from the CPU and therefore, can run in harmony alongside other applications on the system.

We also test the response time of the recommendation system by analyzing the activities of ten different characteristic users (active at the same time) as a result of one and ten content selections as shown in Fig. 4. Because of each individual characteristic, the creating content relations from

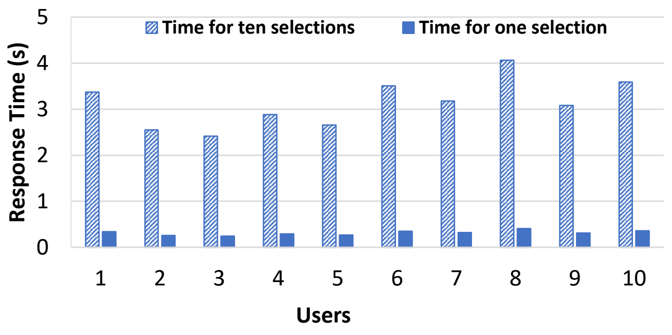

Fig. 4. Response time for the activities of ten active users that have distinct characteristic for one and ten consecutive selections.

the selected options can consume distinct time for each individual. Although, we run 200 realizations, the other running application can also be a reason for different response time for different users. Though small differences, the response times are reasonable for each user because the average response time can rise only upto 0.4s.

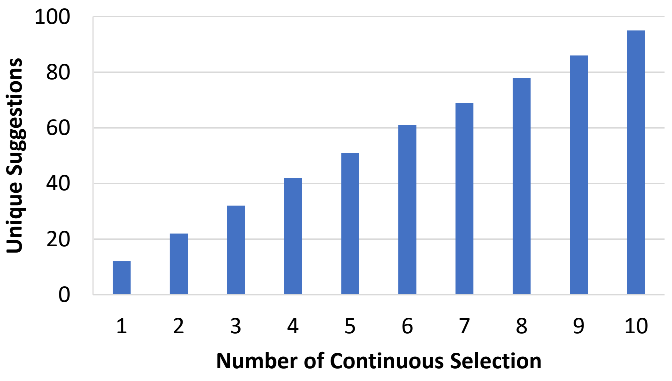

Fig. 5. Suggestion uniqueness comparison of the recommendation systems with and without recommendation improvements according to one to ten content selections out of 100 suggestions.

To test the learning of the system, we test the suggested content uniqueness as a result of selections of simulated users as shown in Fig. 5. In this scenario, we test the system with and without recommendation system improvements. Fig. 5 shows the average number of unique contents for simulated users from one to ten continuous selections comparing to the same recommendation system with/out applying recommendation system improvements. While the number of selections is increased from users, the differences between the recommendation systems also increase. Indeed, this shows that the proposed recommendation system also affects the long-term suggestions.

We also test the recommendation system improvements by only considering the one parameter of aforementioned parameters such as popularity to test the uniqueness of the suggested contents with and without recommendation improvements over time to test the effects of each parameter to the improvements. For this test, we used Popularity, Similarity and Location parameters.

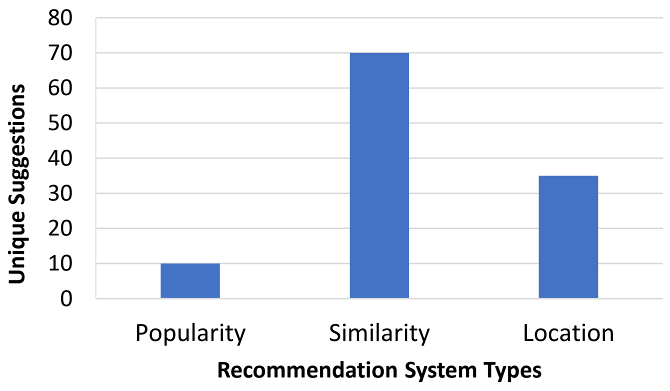

Fig. 6. A number of unique suggestions from Popularity, Similarity, and Location recommendation systems with and without recommendation improvements out of 100 suggestions.

- Popularity mode considers the searched content with popularity and only returns the popular related data of searched contents from the database.
- Similarity mode considers the searched content with the most related contents and only returns the most related contents to the searched contents from the database.
- Location mode only returns the contents from the similar location which users access.

Fig. 6 shows the average number of unique contents for simulated users from ten selections comparing to recommendation systems based on Popularity, Similarity, and Location with/out applying recommendation system improvements. Because of a limited number of popular contents, the uniqueness of the suggested contents in Popularity-based recommendation system is significantly lower than the recommendation systems based on Similarity and Location. On the other hand, Similarity has the largest uniqueness because of larger selections options.

A. Summary of Results

Based on the results, we make the following observations: (i) Recommendation system improvements do not decrease the system performance because of double filtering overheads; (ii) The uniqueness of the suggested contents gradually increases while the users make consecutive selections, and (iii) The recommendation system improvement has different effects on different recommendation systems in terms of uniqueness of suggested contents.

VII. Conclusion and Future Work

In this paper, we proposed the Predictive Self-Learning Recommendation System that uses a Collaborative Filtering Algorithm as well as seven criteria (Popularity, Similarity, Currency, Feedback, Importance, Safety, and Interest) in addition to users' profiles to make predictive recommendations to users. The system is different from traditional recommendation systems because it allows for more diverse suggestions without decreasing the performance of the system in terms of response time and CPU utilization. We conduct an extensive simulation to test the performance the proposed recommendation system in terms of system performance, system efficiency, and system integrity. Our experimental results show that response time can increase upto 0.4s and CPU utilization remains at 20% level. Moreover, the uniqueness of the suggested contents increases while users make selections.

Although, the proposed method and related analysis can assists the shopping, entertainment and similar recommendation systems to increase their efficiency by well-characterizing users' behaviors, the more work needs to be done in terms of testing, used dataset, and the learning methodology. Therefore, in the future, we want to test the proposed method on a commercial website with different types of recommendation systems.

References

[1] C. Goodrow. (2017, Feb.) You know whats cool? A billion hours. YouTube. [Online]. Available: https://youtube.googleblog.com/2017/02/you-know-whats-cool-billion-hours.html

[2] M. Wiesner and D. Pfeifer, "Health recommender systems: concepts, requirements, technical basics and challenges," *International journal of environmental research and public health*, vol. 11, no. 3, pp. 2580–2607, 2014.

[3] B. Smith and G. Linden, "Two decades of recommender systems at amazon. com," *IEEE Internet Computing*, vol. 21, no. 3, pp. 12–18, 2017.

[4] J. A. Konstan and J. Riedl, "Deconstructing recommender systems," IEEE Spectrum: Technology, Engineering, and Science News, 2017, accessed: Sep 27, 2017. [Online]. Available: https://spectrum.ieee.org/computing/software/deconstructing-recommender-systems

[5] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, p. 4, 2009.

[6] N. Lee, J. J. Jung, A. Selamat, and D. Hwang, "Black-box testing of practical movie recommendation systems: A comparative study," *Computer Science and Information Systems*, vol. 11, no. 1, pp. 241–249, 2014.

[7] A. Liang, "Rotten tomatoes: Sentiment classification in movie reviews," *CS*, vol. 229, p. 15, 2006.

[8] J. Beel, M. Genzmehr, S. Langer, A. Nürnberger, and B. Gipp, "A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation," in *Proceedings of the international workshop on reproducibility and replication in recommender systems evaluation*. ACM, 2013, pp. 7–14.

[9] S. Zhang, L. Yao, and A. Sun, "Deep learning based recommender system: A survey and new perspectives," *arXiv preprint arXiv:1707.07435*, 2017.

[10] J. He and W. W. Chu, "A social network-based recommender system (snrs)," in *Data Mining for Social Network Data*. Springer, 2010, pp. 47–74.

[11] D. Asanov *et al.*, "Algorithms and methods in recommender systems," *Berlin Institute of Technology, Berlin, Germany*, 2011.

[12] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston *et al.*, "The youtube video recommendation system," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 293–296.

[13] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Transactions on Management Information Systems (TMIS)*, vol. 6, no. 4, p. 13, 2016.

[14] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, vol. 7, no. 1, pp. 76–80, 2003.

[15] X. Ye, J. Li, Z. Qi, and X. He, "Enhancing retrieval and ranking performance for media search engine by deep learning," in *System Sciences (HICSS), 2016 49th Hawaii International Conference on*. IEEE, 2016, pp. 1174–1180.

[16] A. Ng, "Collaborative filtering," *Stanford University Lecture Notes*, 2017.

[17] [Online]. Available: https://www.youtube.com/