A System for Suggesting Suitable Content to the Users of Sahave

Toqeer Israr, Eastern Illinois University; Ajay Aakula, Eastern Illinois University

Abstract

Sahave, a start-up company in Illinois, develops community service applications, offering three major services: blood donation, volunteering, and campaign activities. Each user is identified by a certain set of characteristics, thereby meeting certain conditions. If a Sahave user, *u*, subscribes to an activity then, as per traditional programming logic, that activity notification gets pushed to the rest of the users, *U*, who display a similar set of characteristics to user *u*. The problem arises when one user in the set *U* is not interested in that activity and still ends up receiving the notification. This spamming of the uninterested users causes discontent and dissatisfaction, and could possibly cause the user to quit. In this study, the authors analyzed user behavior and, in this paper, propose a preference system for Sahave users. In a traditional subscription model, users are notified of events for which they have subscribed, and possibly for others that similar users have opted-in for. In this paper, the authors propose machine learning algorithms to make the Sahave's system smart, by using collaborative filtering, content-based filtering, and hybrid filtering.

Collaborative filtering is based on the idea that people who agreed in their evaluation to certain items in the past are likely to agree in the future. A content-based filtering system works with data that the user provides, either explicitly (providing feedback) or implicitly (clicking on a link). A user profile is created based on user data and is used to make suggestions to the user. The system improves over time by self-training as the user provides more input or takes actions on the recommendations. A hybrid approach combines content-based filtering and collaborative filtering, utilizing the benefits of both. Depending on the problem and situation, Sahave can choose the system that will work best.

Introduction

Sahave is a startup organization in Illinois, working on community service application products. The community service application, also known as *Sahave*, deals with blood donation, volunteering, and campaign creation services (to avoid ambiguity, the company will be referred to as "*company* Sahave" and the application as just "Sahave"). Sahave has built a mobile application using technologies such as Angular JS, Ionic, Node.js, and MongoDB. This application resides on Amazon's web service server. The application has a major feature that notifies users about related events. The current notification system in the application is not intelligent enough to understand user interest before sending a notification. This could cause some problems for users who may perceive these notifications as spam, due to the low-intelligent nature of the application.

Traditional machine learning is defined as, "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E" (Mitchell, 1997). Machine learning can be very powerful with the help of more data, as illustrated in Figure 1. Because of the evolution of big data, due to social media, IoT, etc., there is a surplus of data available to train machine-learning models to perform intelligent operations.

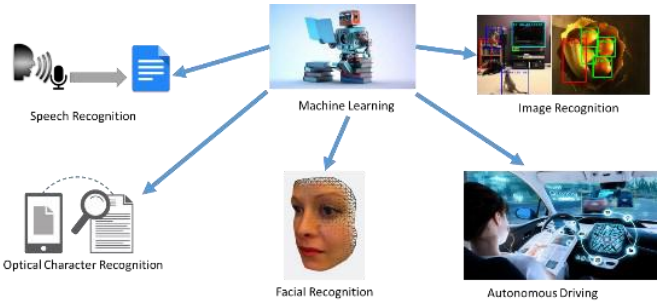

Figure 1. Machine learning application.

Take Netflix as an example. Initially, Netflix was a movie rental website, but with the help of a technology revolution, it started streaming video and on-demand services. As there is a significant number of videos and on-demand services available online, the user faces difficulty in choosing what to watch. Netflix added a recommendation feature with the help of machine-learning techniques, which suggested movies to users based on their viewing history. The Netflix recommendation system is quite intelligent for predicting which movie is best for the user, based on past action data.

Specific Problem

This paper mainly focuses on problems that Sahave users are facing, due to the notification system. The *company* Sahave offers three services: blood donation, volunteering, and campaign services. Blood donation services allow the

administrators to create a blood donation request for collecting blood from donors. Volunteer services allow administrators to create opportunity requests for social service activities for volunteers to sign up. Campaign services allow administrators to run campaign activities for specific causes to provide support for a specifically targeted cause and/or people.

Presently, *company* Sahave has a recommendation notification system, Sahave, which notifies users of interesting and related event details. However, the current system has difficulty in understanding user interests in detail. If the Sahave user has created a campaign about breast cancer and is encouraging women of certain ages to get mammography exams, then most likely male users of the system would not be interested in receiving this notification. However, the current system will push this out to all users (based on the location) who may have subscribed to health-related campaigns. Sahave depends only on broad-user preferences and location details to push such notifications. The blood donation service allows users to participate in blood donation programs. Users can also become administrators and can create their own blood donation activity. Once a user-cum-administrator has created an individual blood donation activity, it then becomes a challenge for Sahave to identify which users should be notified. Figure 2 explains the current Sahave blood donation notification system functionality.

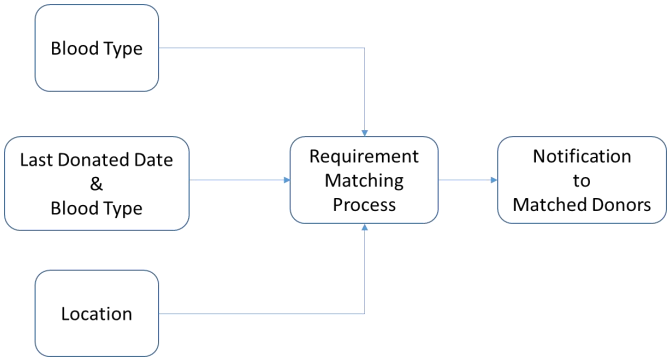

Figure 2. Blood donation service notification system.

Currently, a user-cum-administrator-created activity, Sahave, based on the details of location and blood group of that activity, sends notifications to the users, where user location and blood group details match up. However, it does not take into account the fact that the very same users have not shown prior interest in similar activities. An intelligent system is needed to analyze user profiles with past user data to make smart decisions about which users should be notified of the new event. The current system sends a notification to all matched users irrespective of users past data.

Volunteer service offers a platform to the users to create volunteer activities. In this service, every user has the ability to create events for different service-oriented purposes. Suppose a Sahave user created a spoken English development program activity to help people improve conversational English skills. Figure 3 explains the current Sahave volunteer service notification system functionality.

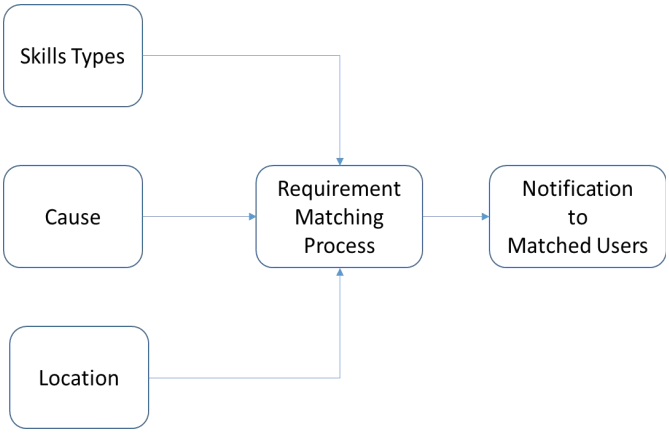

Figure 3. Volunteer service notification system.

The event will be conducted in a particular place at a specific time. After the user has created the event, the current Sahave recommendation system will push the notifications to all of the users located in that particular area with the matched skillset, using some predefined conditions. However, this is not useful for a user already proficient in English. If a similar activity is created again and the same user is spammed again with a notification for an English class event, the user may be extremely unhappy and will most likely either block or unsubscribe from the notification list. A user always expects to receive recommendations that match his or her interests. If the system sends unrelated notifications at regular intervals, then the user will not be happy. The main problem in the current system is that it is too difficult to understand varying user interests. The trouble with the current system is that it is not able to correlate user personal criteria with past activities. If 100 Sahave users create a volunteer event at the same place with the same criteria, then it pushes notifications to all users who meet these criteria 100 times. However, this will cause quite an unpleasant experience for users, especially when they might not be interested in such activities.

Research

The research problem needs to take into account details using different contexts and note that solving this problem using traditional programming languages would be difficult. The recent technology evolution in big data offers support

to intelligently deal with user problems. Machine learning can deal with problems that are difficult to resolve using traditional programming languages.

A machine-learning model was developed using different algorithm functions. Many algorithms can help in developing a machine-learning model to understand user preferences in different dimensions for effective recommendations to the users. But every model requires user "training" data to predict values in a real-time scenario based on such data. The current system can be replaced with the new machine-learning-based recommendation system, which is broadly divided into three approaches, based on user data: collaborative filtering, content-based filtering, and hybrid filtering (Aggarwal, Tomar, & Kathuria, 2017).

Collaborative Filtering

Assumption. Users having a similar interest in the past are most likely to have a similar interest in future.

Collaborative filtering is defined as the system trying to send a notification to the user based on the popularity of the item. Collaborative filtering is classified into two types, based on items and users (Aggarwal, Tomar, & Kathuria, 2017). Figure 4 shows user-based filtering and is based on user reviews such as ratings on specific items. Now suppose that user X is attending three events—a dance class, an English language verbal event, and an English language writing skills event—and user Y is attending two of those three events. In this case, user-based filtering suggests that user Y attend the remaining third event. Here, there is a correlation between user X and user Y, so an algorithm identifies unmatched events that can be recommend to each user. A user-based collaborative filtering system makes the recommendations to users based on user data.

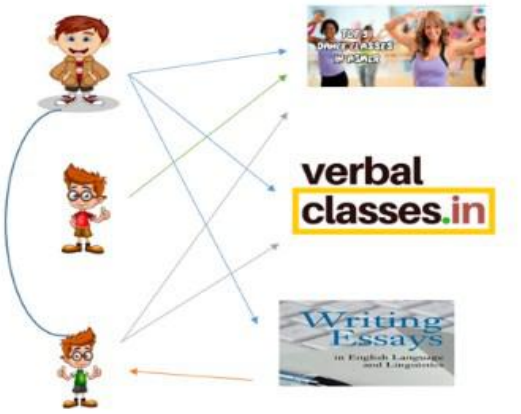

Figure 4. User-based collaborative filtering (UB-CF).

Figure 5 shows item-based collaborative filtering, which mainly explores the relationship between items. If a user attends a particular volunteer event, then he/she may attend other, closely related events. Or, if user X is attending an English language verbal event and many users in the past also attended an English language writing skills event along with the verbal event, then there is a high probability that user X would also like to attend an English language writing skills event (Collaborative, 2012). An item-based collaborative filtering system makes the recommendations based on item data.

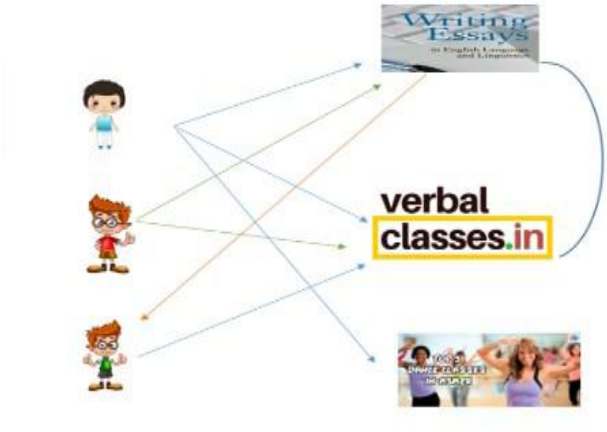

Figure 5. Item-based collaborative filtering (IB-CF).

Here is another example to illustrate how a collaborative filtering system makes recommendations. Table 1 presents five opportunity creation events and five people. A "+" indicates that the person participated in the event, and a "–" indicates that the person did not participate in the event. To predict if Ken would like to attend an event for résumé building, his previously attended events are compared to the events of the other users. In this case, the events of Ken and Mike are identical, and Mike liked résumé building, so everyone can predict that Ken likes the résumé building opportunity as well.

Table 1. User event attendance.

	Amy	Jeff	Mike	Chris	Ken
Cricket	–	–	+		+
Teaching Skills	–	+	+	–	+
Acting	+		–	+	–
Business Skills	–	–	+	–	+
Resume building	–	+	+	–	?

However, the system cannot depend on similar-person criteria every time. Instead of relying on the most similar person, a prediction is usually based on the weighted average of several users' recommendations. How can the system calculate a weight coefficient to assign to user attendance values? The weight given to a person's attendance is determined by the correlation between that person and the person who will make a prediction.

As a measure of correlation, the Pearson correlation coefficient can be used. The Pearson correlation coefficient helps to determine the relationship between two variables; the relationship unit varies from -1 to +1, -1, when there is a perfect negative linear relationship and +1, when there is a perfect positive linear relationship (Sari, Dal'Col Lúcio, Santana, Krysczun, Tischler, & Drebes, 2017). The relationship between two persons/events can be calculated using this coefficient. The attendance of persons X and Y of event k is written as X_k and Y_k, while X! and Y! are the mean values of all their event attendance in the past. The correlation between X and Y is then given by Equation 1 (Collaborative, 2012):

$$r(X,Y) = \frac{\sum (XK - X!)(YK - Y!)}{\sqrt{\sum_k (XK - X!)^2 \sum_k (YK - Y!)^2}} \quad (1)$$

In Equation 1, k is an element of all of the events that both X and Y have attended in the past. A prediction for the event attendance of person X of the event item i, based on the attendance of people who have attended event item i, can be computed using Equation 2 (Collaborative, 2012):

$$P(X_i) = \frac{\sum_K (Y_i) r(X,Y)}{n} \quad (2)$$

where, k consists of all of the people, n, who have attended event item i.

Note that a negative correlation can also be used as a weight. For example, Amy and Jeff have a negative correlation and Amy did not like résumé building, which could be used as an indication that Jeff will enjoy résumé building. If no one has attended event item i, the prediction is equal to the average of all of the events that person X attended. The constrained Pearson measure is similar to the normal Pearson measure but uses the mean value of possible attended events instead of the mean values of the attended events of persons X and Y. The system can make accurate predictions to Sahave users based on a collaborative filtering mechanism.

Content-Based Filtering

Assumption. The key parameters in the content of an item are more closely related to the user experience data than the generic metadata. Hence, this will separate relevant items from non-relevant items.

A content-based recommendation system works with user-provided data. Based on those data, a user profile is generated, which is then used to make suggestions to the user. As the user provides more input or takes action on recommendations, the engine becomes more and more accurate (Aggarwal, Tomar, & Kathuria, 2017). Term frequency (TF) and inverse document frequency (IDF) are used in information retrieval systems as well as content-based filtering mechanisms. They are used to find the comparative importance of a document, article, news item, movie, etc. TF is known as the frequency of a word in a document. IDF is defined as the inverse of the document frequency among the whole corpus of documents (Aggarwal, Tomar, & Kathuria, 2017).

Figure 6 shows the content-based filtering mechanism. As per the content-based filtering approach, this new system creates profiles for each user, based on the user's previous action data. If a user-cum-administrator created two volunteer activities, say, training on Web development and training on writing development, the new system will find which activity is notified based on the word terms in the activity data and user profile data. It is certain that "the" will occur more frequently than "development," but the relative importance of "development" is higher than the more frequent word "the." A TF-IDF weighting technique will negate the effect of high-frequency words in determining the importance of an activity to the user.

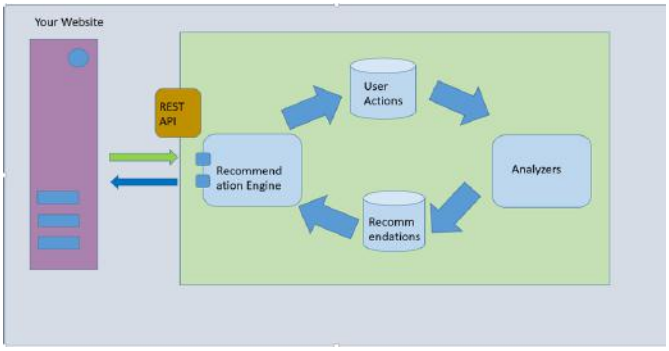

Figure 6. Content-based filtering mechanism.

While calculating a TF-IDF, many unimportant high-frequency words, such as "the," "and," "more," etc. can be identified. The log is then used to dampen the effect of high

-frequency words, as indicated by Equation 3. For example, TF = 3 vs TF = 4 is vastly different from TF = 10 vs TF = 1000. The importance of a word cannot be calculated using simple word count in an item (Aggarwal, Tomar, & Kathuria, 2017). Table 2 indicates that the effect of high-frequency words is dampened, and new values are more comparable to each other, as opposed to the original raw-term frequency.

$$\begin{cases} 1 + log_{10}\, tf_{t,d}, if\ tf_{t,d} > 10 \\ 0, otherwise \end{cases} \quad (3)$$

Table 2. Term frequency and weighted term frequency values.

Term Frequency	Weighted Term Frequency
0	0
10	2
1000	4

The TF-IDF Method

Eliminate stop words—Example: "in," "the," "so," "but," etc. Find the importance of a word's measure by calculating the TF.IDF score (term frequency multiplied by inverse document frequency).

Term frequency—Word count in the content:

TF(t) = (Number of times the term *t* appears in the document) / (Total number of terms in the document)
IDF(t) = loge(Total number of documents / Number of documents with the term *t* in it)

Assume that two events and a user profile contain the words listed in Table 3, then calculate term frequency and inverse document frequency.

How the Vector Space Model Works

TF (training, v1) = 3/6 = 0.5
TF (training, v2) = 1/4 = 0.25
TF (user, *u*) = 1/6 = 0.16
IDF (training) = log (3/3) = 0
TF.IDF (training, v1) = 0*0.5 = 0
TF.IDF (training, v2) = 0*0.25 = 0
TF.IDF (user, *u*) = 0*0.16 = 0

TF (Web, v1) = 2/6 = 0.33
TF (Web, v2) = 0/4 = 0
TF (user, *u*) = 2/6 = 0.66
IDF (Web) = log (3/2) = 0.17
TF.IDF (Web, v1) = 0.33*0.17 = 0.22
TF. IDF (Web, v2) = 0.17*0 = 0
TF. IDF (user, *u*) = 0.17*0.66 = 0.11

TF (writing, v1) = 0/6 = 0
TF (writing, v2) = 1/4 = 0.25
TF (user, *u*) = 1/6 = 0.16
IDF (writing) = log (3/2) = 0.17
TF.IDF (writing, v1) = 0.17*0 = 0
TF. IDF (writing, v2) = 0.17*0.25 = 0.04
TF. IDF (writing, *u*) = 0.17*0.16 = 0.02

TF (Development, v1) = 1/6 = 0.16
TF (Development, v2) = 2/4 = 0.5
TF (Development, *u*) = 2/6 = 0.66
IDF (Development, *V*) = log (3/3) = 0
TF.IDF (Development, v1) = 0.16*0 = 0
TF. IDF (Development, v2) = 0.5*0 = 0
TF. IDF (Development, v2) = 0.66*0 = 0

Table 3. User profile, volunteer events 1 and 2, and word count.

Volunteer event 1(v1)	Word	Count
	Training	3
	Web	2
	Development	1
Volunteer event 2(v2)	Word	Count
	Training	1
	Writing	1
	Development	2
User profile(u)	Word	Count
	Training	1
	Writing	1
	Development	2
	Web	2

After calculating TF-IDF scores for the user profile, and volunteer events as shown in Table 4, the system can determine which event is closer to the user profile. This is accomplished using the vector space model, which computes the nearness, based on the angle between the vectors. A vector space model is algebraic, which helps in filtering information and relevancy ranking (Arguello, 2013). A vector is a point in the vector space; thus, calculating the similarity between two items is required. The items need to be stored as vectors of attributes in *n*-dimensional space.

Table 4. User profile and volunteer events 1 and 2.

S.no	Web	Training	Development
Volunteer event 1	0.66	0	0
Volunteer event 2	0	0	0.33
User profile	0.11	0	0

Figure 7 represents the three-dimensional vector space for simplicity, which is defined as a basis vector with "training web development" attributes. Figure 7 is a 3D representation of three attributes: development, Web, and training. The vector space model ranks the event based on vector space similarity between volunteer activity events and user profiles. The system performs a cosine similarity between the user profile vector and volunteer activity vector then calculates the angle between the vectors. From Equation 4, the ultimate reason for using cosine is that the value of cosine will increase with a decreasing value of the angle, which signifies more similarity. The angle between the user profile vector and volunteer activity 1 is 0, and the angle between the user profile vector and volunteer activity 2 is 90. Hence, volunteer activity 1 is closer to the user profile.

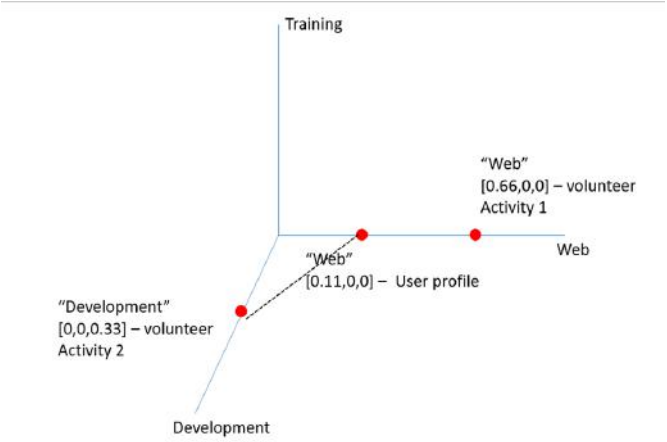

Figure 7. Vector space model.

$$\cos\theta = \frac{volunteer\ activity1\ .user\ profile}{volunteer\ activity1\ user\ profile} \quad (4)$$

Hybrid Filtering

Many times, there will be scenarios for which either content-based filtering or collaborative filtering may not apply completely, or both of them can be used. For these scenarios, the recommendation system uses hybrid filtering. Both content-based filtering and collaborative filtering have their advantages and disadvantages. The following specific problems can be distinguished for content-based filtering.

- Content description: It is difficult to generate a meaningful content description in every area of business (Hybrid, 2012).
- Over-specialization: A content-based filtering system could not recommend items if the last user's behavior does not provide evidence for this. Additional techniques must be added to allow the system the capability to make a recommendation that is not within the scope of what the user has already shown interest in (Hybrid, 2012).

A collaborative filtering system does not have these weaknesses, because it does not require content in order to recommend an item to a user. The system can handle any kind of information. Furthermore, the system can recommend items to a user that may have very different content from what the user has previously indicated interest in. Finally, because recommendations are based on the opinions of others, it is well suited for subjective domains like art. However, collaborative filtering does introduce certain problems of its own:

- Early-rater problem: Collaborative filtering systems are not able to recommend new items to users, since they do not have user ratings on which to make predictions. Even if users start participating in events, it will take some time before the events have received enough of a participation count to make accurate recommendations. Similarly, recommendations will also be inaccurate for new users only having attended a few events (Hybrid, 2012).
- Sparsity problem: In many information domains, the items have more data for a person that they can discover. This makes it hard to find events that are attended by enough people on which to base predictions (Hybrid, 2012).
- Gray sheep: In general, user groups with overlapping characteristics are needed. Even if such groups exist, individuals may not be able to agree or disagree with any group of people and will receive incorrect recommendations (Hybrid, 21012).

A hybrid filtering system is a combination of content-based filtering and collaborative filtering. This system could take advantage of both the representation of the content as well as the similarities among users. In a hybrid approach, two types of information need to be combined for the recommendation, and it is possible to use the recommendations of the two filtering techniques independently (Hybrid, 2012).

Collaborative filtering is based on the correlation between users to make predictions. Such a correlation is most meaningful and accurate when users have attended many events in common. Furthermore, the lack of access to the content of the items prevents similar users from being matched, unless they have attended the exact same event. For example, if one user liked the event "listening skills" and another liked the event "writing skills," they would not necessarily be matched together. A hybrid approach called collaboration via content deals with these issues by incorporating both the information used by content-based filtering and collaborative filtering. In collaboration via content, both the attendance at events and the content of the events are used to construct a user profile. The selection of terms that describe the content of the events is done using content-based techniques. The weight of terms indicates their importance for the user. Table 5 shows an example of the kind of information available for making a prediction about the event résumé building for Ken with collaboration via content.

Table 5. User attendance and event content scores.

	Experienced	Software Engineer	Fresher	Host	Hyderabad	Résumé Building
Amy	1	0	1.2	0.2	0.2	–
Jeff	2.1	0	0.5	3	2.2	+
Mike	1.3	1.5	0.2	3.2	1.9	+
Chris	1.1	2	2.8	0.8	0	–
Ken	0.8	1.1	0	2	1.2	?

Just as with collaborative filtering, the Pearson correlation coefficient can be used to compute the correlation between users. Instead of determining the correlation with a user who attended the events, however, term weights are used. Because this method has a greater number of items from which to determine similarity than collaborative filtering, the problem of users not having enough common events attended is no longer an issue. Furthermore, unlike content-based filtering, predictions are based on the impressions of other users, which could lead to recommendations outside the normal environment of a user. However, to make recommendations about events, it is still necessary that enough users attend the event. Just as with collaborative filtering, new events cannot be recommended if no user has attended the events.

Another approach to combining collaborative and content -based filtering is to make predictions based on a weighted average of the content-based recommendation and the collaborative recommendation. The rank of each item being recommended could be a measure of the weight. In this way, the highest recommendation receives the highest weights.

Discussion

This study shares the most useful techniques for utilization in the Sahave application for the effective recommendation engine. Both collaborative and content-based filtering approaches have their own ways of resolving the issue, but using a combination of both approaches can address the issue in a different context. Applications cannot blindly rely on these systems, because both have limitations. If the application does not have sufficient data, users cannot expect accurate recommendations. If the application has fewer users, then it would be good to have a traditional recommendation system engine instead of developing one that is based on machine learning. Users can expect good recommendations from the system recommended from this study, if and only if the user and item counts are relatively high. If user access is minimal, then there might be a chance of inaccurate recommendations getting pushed to the users, due to inconsistent user data.

Conclusions and Future Work

Sahave is an application for community work such as blood donation, volunteer activities, and managing campaigns. It currently faces problems when it tries to notify its users of new events, sometimes resulting in unwanted/ unneeded notifications. Overall, these problems can be considerably decreased by using the methods discussed in this paper. Sometimes it is difficult to recommend exact user interests every time. If the system does not have a good amount of user data, then it is difficult to predict user interest accurately. In this paper, the authors discussed three models for resolving user problems. Many organizations prefer to implement a hybrid filtering approach, as it combines both content-based and collaborative filtering approaches. Sahave has not yet implemented a recommendation system using machine-learning techniques. They have just recently launched the Sahave application. Once a relatively large amount of data is generated, then they make use of the system recommended in this study.

References

Aggarwal, P., Tomar, V., & Kathuria, A. (2017). Comparing content based and collaborative filtering in recommender systems. *International Journal of New Technology and Research*, *3*(4), 65-67. Retrieved from https://www.ijntr.org/download_data/IJNTR03040022.pdf

Arguello, J. (2013, February 13). *Vector space model*. [PowerPoint slides]. Retrieved from https://ils.unc.edu/courses/2013_spring/inls509_001/lectures/06-VectorSpaceModel.pdf

Collaborative filtering. (2012, January 23). Retrieved from http://recommender-systems.org/collaborative-filtering/

Hybrid recommender systems. (2012, January 22). Retrieved from http://recommender-systems.org/hybrid-recommender-systems/

Mitchell, T. M. (1997). *Machine learning*. New York: McGraw-Hill.

Sari, B. G., Dal'Col Lúcio, A., Santana, C. S., Krysczun, D. K., Tischler, A. L., & Drebes, L. (2017). Sample size for estimation of the Pearson correlation coefficient in cherry tomato tests. *Ciência Rural*, *47*(10), 1–6. doi: http://dx.doi.org/10.1590/0103-8478cr20170116

Biographies

TOQEER ISRAR is a Professional Engineer and an assistant professor at Eastern Illinois University. He earned his PhD in electrical and computer engineering, 2014, from the University of Ottawa, Ontario, Canada, and his BEng and MASc in electrical engineering degrees from Carleton University, Ontario, Canada. He is an internationally recognized expert in the areas of performance and software engineering, and has several years' of experience in industry. Dr. Israr may be reached at taisrar@eiu.edu

AJAY AAKULA is a student at Eastern Illinois University. He holds a Bachelor of Technology degree in Electronics and Communication Engineering from Jawaharlal Nehru Technological University, Hyderabad, India. Currently, he is pursuing a master's program in computer technology at Eastern Illinois University. He has over three years' of experience as an application developer, and he has worked in banking and healthcare. Mr. Aakula may be reached at aakula@eiu.edu