Recommendation System for Suggesting Suitable Content to the Users of Sahave

Toqeer Israr
Eastern Illinois University
taisrar@eiu.edu

Ajay Aakula
Eastern Illinois University
aakula@eiu.edu

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 and thus meets certain conditions. If a Sahave user, *u*, subscribes to an activity, then as per traditional programming logic, that activity notification gets pushed to the other users, U, who also displays similiar set of characteristics to those of 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. Spamming uninterested users causes discontent and dissatisfaction and could possibly cause a user to quit.

This paper analyzes user behavior and proposes a future preference for Sahave users. In a traditional subscription model, users will be notified of events for which they have subscribed to and possibly to those that other "similar" users have opted-in in. We propose machine-learning algorithms to make the Sahave's system smart: collaborative filtering, content-based recommedation system, and a hybrid.

Collaborative filtering functioning is based on the idea that people who agreed in their past evaluation of certain items are likely to agree in the future. A content-based recommendation 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 action on the recommendations. A hybrid approach combines content-based filtering and collaborative filtering, using 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, we will refer to the company as "*company* Sahave" and the application as just "Sahave"). It has built a mobile application using technologies such as Angular JS, Ionic, Node.js, and MongoDB.

This application resides in Amazon's Web service server. The application has a major feature that notifies users about related events. The current notification system 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 poor intelligent nature of the application.

In traditional machine learning, "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 is 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., a surplus of data is available to train machine-learning models to perform intelligent operations.

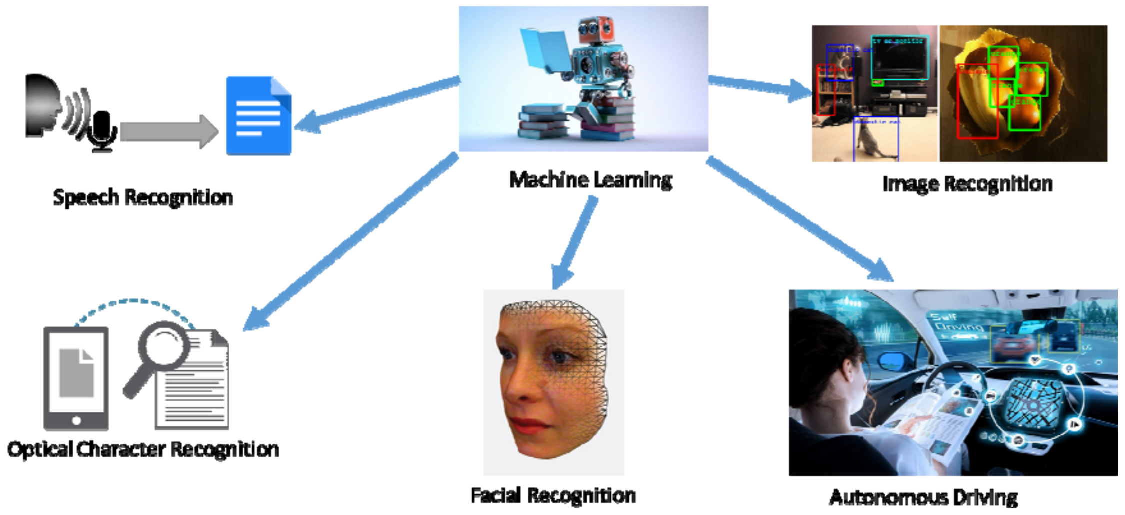

Figure 1. Machine learning application.

Let us consider 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 faced 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 that user's viewing history. With user past action data, the Netflix recommendation system is intelligent enough to predict which movie is best.

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 allows administrators to create a blood donation request for collecting blood from donors. Volunteering allow administrators to create opportunity requests for social service activities. Campaign service allows administrators to run campaign activities for specific causes to support a targeted cause and/or people.

Presently, *company* Sahave has a recommendation notification system, *Sahave*, which notifies users of “interested” and “related” events details. However, the current system has difficulty in understanding detailed user interests. If a Sahave user has 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 donor programs. Users can also become administrators and can create their own blood donation activity. Once if a *user-cum-administor* has created an individual blood donation activity, then it becomes a challenge for *Sahave* to identify which users should be notified, as shown in Figure 2.

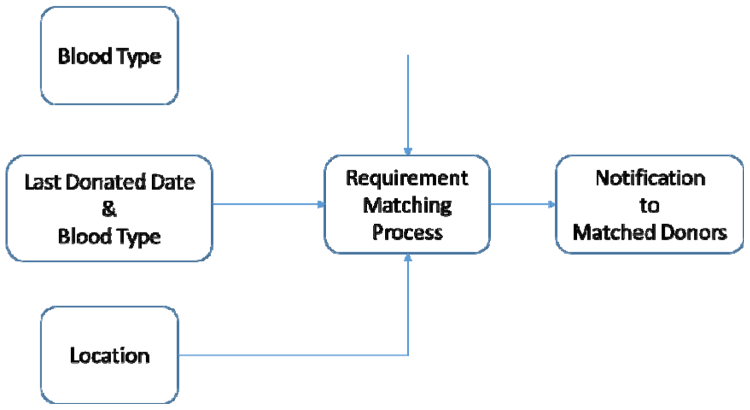
Blood Type
Last Donated Date & Blood Type
Location
Requirement Matching Process
Notification to Matched Donors

Figure 2. Blood donation service notification system.

Currently, a *user-cum-administor* 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 consider 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 a new event. The current system sends a notification to all matched users, irrespective of users’ past data.

Volunteering offers a platform for the users to create volunteer activities. In this service, every user has access to create events for different service-oriented purposes. Let us suppose a Sahave user created a spoken English development program to help people improve conversational English skills, as shown in Figure 3.

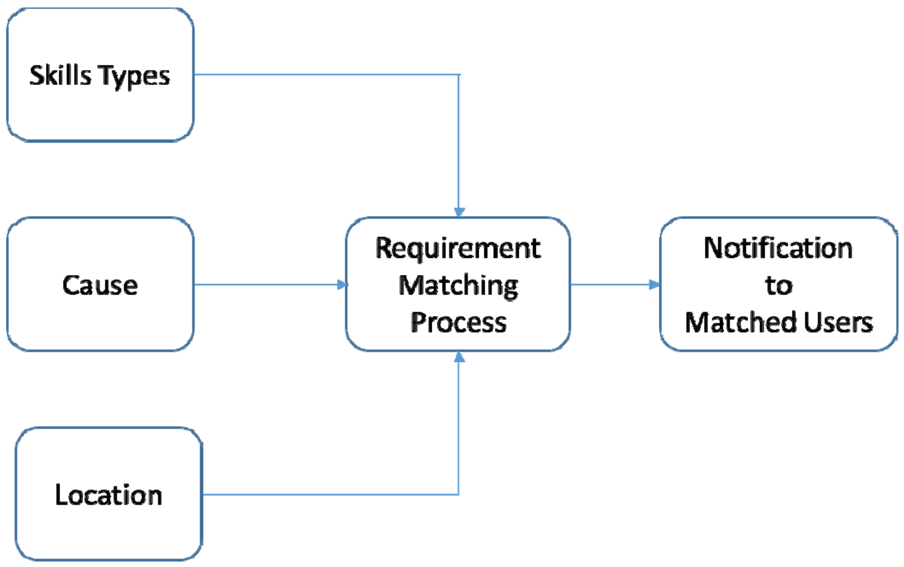

Figure 3. Volunteer service notification system.

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 notifications to all users located in that particular area with the matched skillset, using some predefined conditions. However, this is not useful for a user already proficient at English. If a similar activity is created again and that user is again spammed with a notification for an English class, the user will be extremely unhappy and will most likely will either block or unsubscribe to the notification list.

Users always expect to receive recommendations that match their 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 has difficulty understanding varying user interests. It cannot correlate user personal criteria with past activities. If 100 Sahave users create a volunteer event at the same place with same criteria, then it pushes 100 notifications to all users who meet these criteria. However, this could cause quite an unpleasant experience for users, especially when they might not be interested in such activities.

Research

The authors understand the problem in detail using different contexts and that solving it with traditional programming languages would be difficult, requiring an in-depth understanding of user preferences. However, 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.

Many algorithms can help in developing a machine-learning model to understand the user preferences in different dimensions for more effective recommendations. Every model requires user “training” data to predict values in a real-time scenario based on such data. We can replace the current system with a new machine-learning-based recommendation system. The system is broadly divided into three different filtering approaches based on user data: collaborative, content-based, and hybrid (Aggarwal, Tomar, & Kathuria, 2017).

Collaborative Filtering

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

Collaborative filtering is defined as the system is trying to send a notification to the user based on the popularity of the item (Aggarwal, Tomar, & Kathuria, 2017). Collaborative filtering is classified into two types based on items and users.

User-based filtering, as shown in Figure 4, is based on user reviews such as ratings on specific items. Let us suppose user X is attending three events, such as a dance class, English language verbal skills, and English language writing skills, and user Y is attending two of those three events. In this case, user-based filtering will suggest 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 recommend each other. A user-based collaborative filtering system makes the recommendations based on user data.

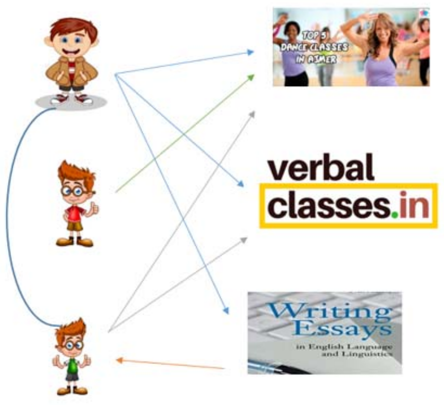
TOP 5 DANCE CLASSES IN AJMER
verbal classes.in
Writing Essays in English Language and Linguistics

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

Item-based collaborative filtering, shown in Figure 5, has mainly explored the relationship between items. If a user attends a particular volunteer event, then s/he may attend other, closely related events. Or if user X is attending an English language verbal event and many users in 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). Item-based collaborative filtering system makes the recommendations based on item data.

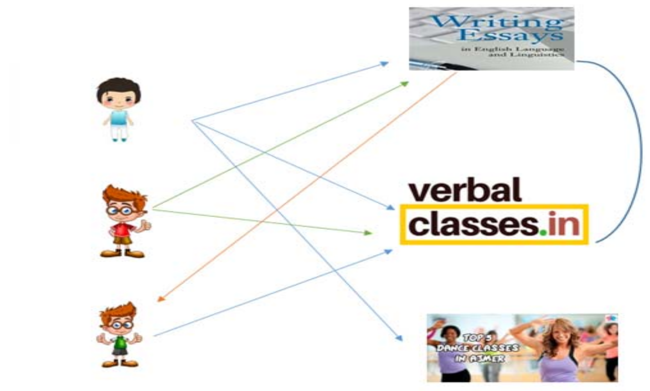
Writing Essays
in English Language and Linguistics
verbal
classes.in
TOP 5 DANCE CLASSES IN AJMER

Figure 5. User-based collaborative filtering (UB-CF).

Let us take 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. To predict if Ken would like to attend the event "resume building," his past-attended events are compared to the events of the other persons. In this case, the events of Ken and Mike are identical, and Mike liked "resume building," so everyone can predict that Ken would like the "resume building" opportunity as well.

Table 1. User event attendance.

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

However, we 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 we 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 for whom to 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 relation unit varies from -1 to +1, -1, when there is a perfect negative linear relation and +1, when there is a perfect positive linear relation (Sari, Dal'Col Lúcio, Santana, Krysczun, Tischler, & Drebes, 2017). We can calculate the relation between two persons/events using this coefficient. Persons X and Y's attendance 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 in Equation 1 (Collaborative, 2012):

$$r(X,Y) = \frac{\sum (X_K - X!)\,(Y_K - Y!)}{\sqrt{\Sigma_k (X_K - X!)^2 \; \Sigma_k (Y_K - Y!)^2}} \qquad (1)$$

In Equation 1, k is an element of all the items that both X and Y have attended an event. A prediction for the event attendance of person X of the item I, based on the attendance of people who have attended item I, is computed as follows in Equation 2 (Collaborative, 2012):

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

Where K consists of all the people who have attended the 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 if Amy did not like "resume building," that could be used as an indication that Jeff will enjoy "resume building." If no one has attended the event item *i*, the prediction is equal to the average of all the attended events of person X. 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 person X and Y. We can make accurate predictions to the 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 the 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).

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 like "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.

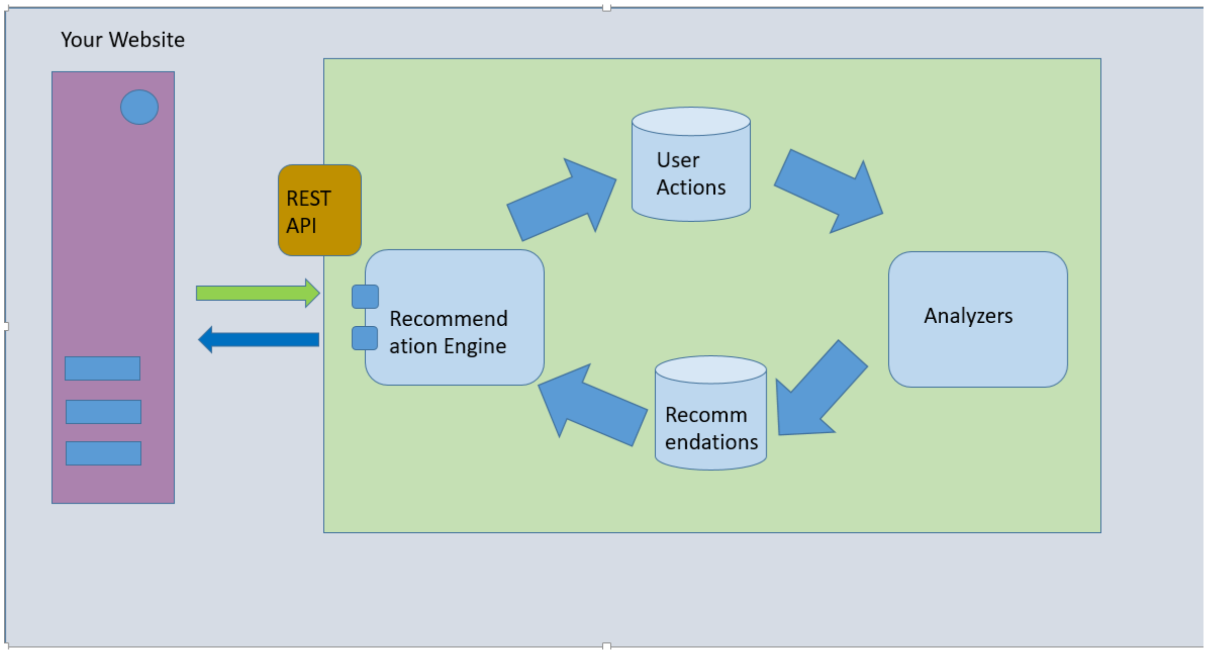

Figure 6. Content-based filtering mechanism.

While calculating TF-IDF, we can identify many unimportant high-frequency words, such as "the," "and," "more," etc., and the log is used to dampen the effect of high-frequency words. 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).

$$\begin{cases} 1 + log_{10}\ tf_{t.d}, & if\ \ tf_{t.d} > 10 \\ 0, & otherwise \end{cases} \tag{3}$$

Table 2. Term frequency and weighted term frequency values.

Term Frequency	**Weighted Term Frequency**
0	0
10	2
1000	4

We can see in Table 2 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.

The TF-IDF Method

1. Eliminate stop words—Example – “in,” “the,” “so,” “but,” etc.
2. The importance of a word measure by calculating TF.IDF score (term frequency multiplied by inverse document frequency)

Term frequency-word count in the content:

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

Let us assume that two documents contain the following words in Table 3, and we can calculate term frequency and inverse document frequency as below:

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 event 1 and 2, 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 of the user profile and volunteer events, we 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.

The Vector Space Model and How It Works

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. We require calculating the similarity between two items. The items need to be stored as vectors of attributes in n-dimensional space.

Figure 7 represents the three-dimensional vector space for simplicity, which is defined as a basis vector with “training Web development” attributes.

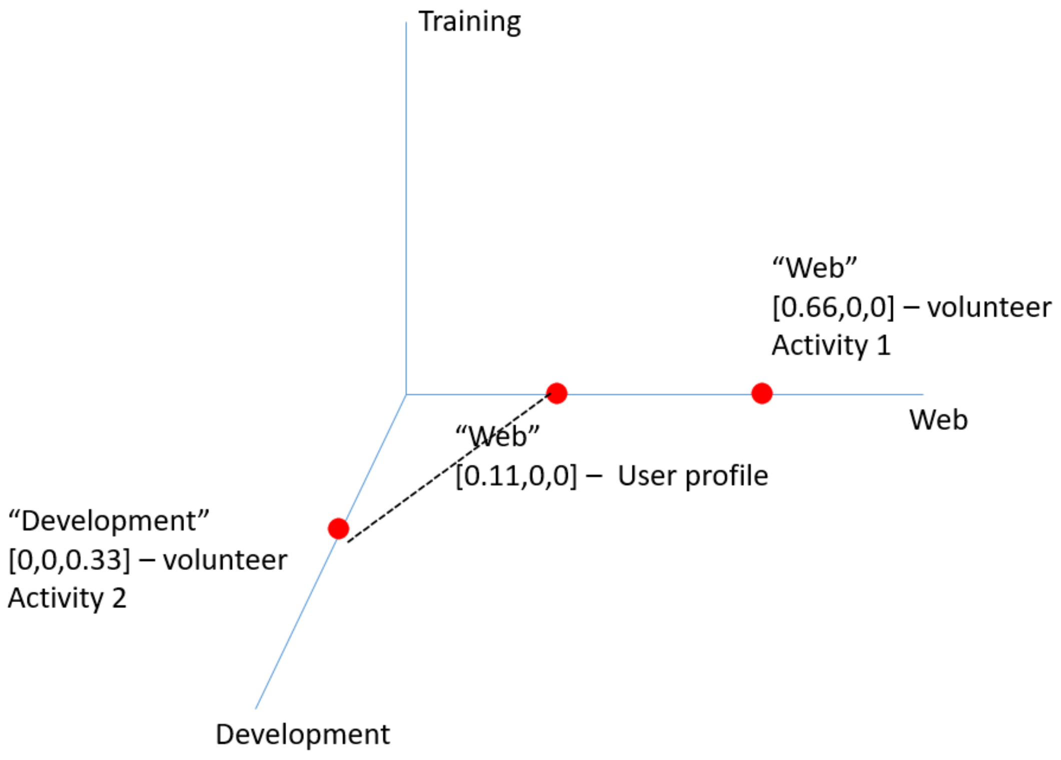

Figure 7. Vector space model.

Table 3. User profile, Volunteer event 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 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. 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.

$$\cos\theta = \frac{volunteer\ activity1\ .user\ profile}{\|volunteer\ activity1\ \|\|user\ profile\ \|}$$

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, we use hybrid filtering. Both content-based filtering and collaborative filtering have their positives and negatives.

The following specific problems can be distinguished for content-based filtering.

- Content description: It is difficult to generate 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 behavior does not provide evidence for this. Additional techniques must be added to make 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 to recommend an item to the user. The system can handle any kind of information. Furthermore, the system can recommend items to the user that may have very different content from what the user has previously indicated interest. Finally, because recommendations are based on the opinions of others, it is well suited for subjective domains like art. However, collaborative filtering introduces certain problems of its own:

- Early rater problem: Collaborative filtering systems are not able to recommend new items to the users since they do not have user ratings to predict. Even if users start participating in events, it will take some time before the events have received enough participation count to make accurate recommendations. Similarly, recommendations will also be inaccurate for new users who have only 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 not be able to agree or disagree with any group of people 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 combine 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 correlation is most meaningful and accurate when users have attended many events in common. Furthermore, 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 by collaborative filtering.

In collaboration via content, both the attendance and content of the events are used to construct a user profile. The selection of terms that describe the content is done using content-based techniques. The weight of terms indicates their importance for the user. Table 4 shows an example the kind of information available to make a prediction about the event "resume building" for Ken with collaboration via content.

Table 4. User attendance and events content score.

	Experienced	Software Engineer	Fresher	Host	Hyderabad	Resume 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	?

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 attended enough common events 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 them. As with collaborative filtering, new events cannot be recommended if no user has attended them.

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 research shares the most useful techniques to utilize in the Sahave application for the effective recommendation engine. Both collaborative and content-based filtering approaches have their own ways to resolving the issue, but using a combination of both approaches can address the issue in a different context. We cannot blindly rely on these systems because both have limitations. If the application does not have sufficient data, we cannot expect accurate recommendations.

If we have a fewer number of user counts, then it would be good to have a traditional recommendation system engine instead of developing one that is machine-learning-based. We can expect a good performance from the above recommended systems 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 being pushed due to inconsistent user data.

Conclusion 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 that we have discussed in this paper. Sometimes it is difficult to recommend exact user interest every time. If the system does not have a good amount of user data, then it is difficult to predict user interest accurately. We have discussed three different types of models to resolve user problems. Many organizations prefers to implement 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 above recommended systems.

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 is currently an assistant professor at Eastern Illinois University. He earned his PhD in Electrical and Computer Engineering, 2014, from University of Ottawa, Ontario, Canada and his BEng and MASc in Electrical Engineering degree from Carleton University, Ontario, Canada. He is an internationally recognized expert in the areas of performance and software engineering and has more than five years of experience in the industry. Dr. Israr may be reached at taisrar@eiu.edu.

AJAY AAKULA is currently a student at Eastern Illinois University. He holds a Bachelor 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, Charleston, IL. He has more than three years of experience as an application developer, and he has worked in banking and healthcare domain. Ajay is available at aakula@eiu.edu.