2018 IEEE International Conference on Data Mining Workshops (ICDMW)

A Scalable Clustering Algorithm for Serendipity in Recommender Systems

Anup Anand Deshmukh*†
Deshmukh.Anand@iiitb.org

Pratheeksha Nair*†
Pratheeksha.Nair@iiitb.org

Shrisha Rao*
shrao@ieee.org

***Abstract*—High sparsity and the problem of overspecialization are challenges faced by collaborative filtering (CF) algorithms in recommender systems. In this paper, we design an approach that efficiently tackles the above problems. We address the first issue of high sparsity in CF by modifying the popular parallel seeding technique proposed by Bahmani *et al.* for use in a spherical setting, for the clustering of users. Experimental evaluations on highly sparse real world datasets demonstrate the better performance of our algorithm than existing Spherical K-Means algorithms. Contrary to the common belief that users are only interested in items similar to those of their previous liking, it has been well established that including serendipitous recommendations improves their satisfaction. Thus, to tackle the second problem of overspecialization, we effectuate serendipity in movie recommender systems with an end-to-end algorithm, Serendipitous Clustering for Collaborative Filtering (SC-CF) that considers diversity, unexpectedness and relevance. SC-CF takes advantage of carefully generated user profiles to then assign users to a "serendipitous cluster." The overall clustering process leverages these user profiles and improves recommendation quality through serendipity. A series of experiments on Serendipity 2018 (part of Movielens) dataset built on real user feedback has shown that SC-CF outperforms the existing popular recommendation methods like K-Means and deep learning based CF.**

***Index Terms*—Serendipity, Clustering, Spherical K-Means, Collaborative filtering**

I. Introduction

Due to the large amount of information available in the online world today, there is need for systematic and organized platforms that provide users with potentially interesting information such as recommendations of movies, books, music, etc. Such platforms, called recommender systems, use intelligent computing techniques taking direct or indirect user feedback and filter useful information. They find suitable items for users based on their preferences, experiences and demographic information [29]. A high quality recommender system should not only capture the preferences of users while taking their feedback but also improve user satisfaction by recommending unsearched yet useful pieces of information.

Among different techniques used by recommender systems, collaborative filtering (CF) is one of the most popular. Collaborative recommenders gather ratings of objects by users and store them in a database [32]. While providing recommendations for a user, a subset of similar users is computed. The similarity between users is calculated based on similar ratings given by them to objects. The system then recommends to a user those objects liked by the subset of similar users. It is to be noted that any given user rates a very small fraction of the entire set of items available for rating. Therefore, this rating information for a user is highly sparse. In this paper, we discuss a modified clustering technique (in Section III-A) that addresses and exploits the sparsity of data.

The evaluation of such recommender systems is often in terms of accuracy which measures how items correspond to the users' rating information [34]. So in order to improve accuracy, most recommender systems recommend only those items that are highly similar to items rated highly by a user previously. Such accuracy-based algorithms thus limit the number of items that can be recommended to a user and causes the problem of *overspecialization* which lowers user satisfaction [34] [6]. To overcome this problem and broaden user preferences, recommender systems should suggest serendipitous items [22]. Our work introduces a clustering-based algorithm, examined in Section III-B, that recommends serendipitous items to users.

However, difficulty in investigating serendipity arises due to the lack of current consensus on the very definition of this concept in recommender systems [25]. There are multiple aspects of serendipity studied in the domain of recommender systems. A study conducted by Kotkov *et al.* [22] reviewing different definitions of serendipity states that a key condition for serendipity is *focus shift*. This occurs when an otherwise uninteresting item becomes interesting. In order to capture this concept, our work examines three aspects of serendipity [22]: *unexpectedness*, *relevance* and *diversity*. For the purpose of our work, we describe these terms as follows.

- Unexpectedness is the difference between recommendations made before and after considering serendipity. It takes into account those items which are not expected to be recommended to the user. (If the user expects the recommendation, then the very meaning of serendipity is lost). Here we assume that the list of recommendations generated before serendipity by a good recommendation algorithm consists of relevant items similar to the user's interests. Unexpected recommendations include items with lower similarity compared to previously-liked items.
- An item is deemed relevant if it is not completely dissimilar from a user's interests and ensures that the serendipitous recommendations are of some value. For this reason, recommendations cannot be chosen at random and there is a need for careful introduction of serendipity.

This work was supported by an Amazon AWS Machine Learning Award.
*International Institute of Information Technology - Bangalore, India
†These authors equally contributed to the paper.

2375-9259/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDMW.2018.00182

IEEE computer society

- Diversity captures the differences among items recommended to users. It is based on the fact that users may have several interests other than those previously liked by them [37]. It is important for broadening user preferences which may affect user satisfaction. In our work, diversity is handled by including item features different from those previously liked by a user.

Thus, a recommendation for a particular user can be termed serendipitous if: (a) it is unexpected by the user; (b) it is relevant to the user; and (c) it broadens the user's preferences by adding diversity to the list of recommended items. In our work, we focus on movie recommender systems and generate recommendations by finding users with similar interests. This we do while integrating the above three aspects of serendipity with the popular method of collaborative filtering.

Our contribution is three-fold and as follows.

1) *Using the SPKM || algorithm (Spherical K-Means parallel) (Algorithm 1)* for clustering users in collaborative filtering. Bahmani *et al.* [5] introduced the K-Means parallel (also called scalable K-Means) clustering algorithm. The seeding technique used in this algorithm has been proved to give good clustering results for a small, constant number of iterations. We use this seeding technique in the spherical setting and thus introduce SPKM ||. In our algorithm (SPKM ||), the metric used for measuring closeness is *cosine similarity* as this can exploit the sparsity of data and converge quickly to a local minima [10]. We also show the scalability of this algorithm by experimenting on two real world datasets.
2) *Introducing the SC-CF algorithm (Serendipitous Clustering for Collaborative Filtering) (Algorithm 2)* for generating recommendations which effectuate serendipity. This is an end-to-end clustering-based algorithm where each user is assigned to multiple clusters—one based on the conventional similarity of user ratings and the other "serendipitous clusters." (Section III-B) The implementation of this algorithm is open-sourced[†].
3) *Validating the SC-CF algorithm on the Serendipity 2018 dataset* [21] which contains real user feedback in the form of ratings of movies recommended to them, including serendipitous recommendations. We also validate the clustering quality by evaluating the SPKM || algorithm on real world datasets for clustering (KOS and ENRON). We use the metrics of root mean square error (RMSE) and mean average error (MAE) to evaluate the SC-CF algorithm. We show that our algorithm outperforms state-of-art algorithms like K-Means and deep learning based CF. Additionally, we evaluate how diverse and unexpected the recommendations generated by SC-CF are using the two metrics, diversity and unexpectedness.

The rest of the paper is organized as follows. Section II discusses literature related to our work. Section III explains the working of the SC-CF algorithm in detail. Section III-A describes the clustering part of the algorithm and Section III-B details the serendipity part. Section IV analyses the dataset and methodology of experiments. In Section V, we present the experimental results obtained by evaluating the SPKM || and SC-CF algorithms. Section VI concludes.

[†]https://github.com/Prat-123/Serendipitous-Clustering-for-Collaborative-Filtering

II. Related Work

Collaborative filtering techniques often identify items relevant to users by correlating them with similar users. The similarity between users can be based on various factors and there are different methods that explore multiple approaches. The Spherical K-Means clustering method is one such approach used by Ampazis [2]. Work by Narang *et al.* [27] also uses similar methods for improving clustering in CF. Methods like the SVD++ algorithm introduced by Kumar *et al.* [23] take different approaches and hypothesize on the importance of opinions, the recommendations of neighborhoods and considers social interaction among users. In order to improve accuracy and scalability of recommendations, the social popularity factor is incorporated in SVD++ factorization method as implicit feedback. In BiasedMF [24], Wang *et al.* combine contextual content of item via a function to produce the context influence factor. They also introduce a vector space embedding method that vectorizes users and items before passing them through a Deep Neural Network (DNN) [36] pipeline to handle the sparsity of data. Chen *et al.* introduce an evolutionary heterogeneous clustering for collaborative filtering (EHC-CF) [38] where user-based clustering occurs based on stable states. Their proposed method is more efficient than k-Means algorithm in a way that it does not require the predefined number of clusters.

Spink *et al.* [31] discovered that *partially relevant* results containing multiple concepts are valuable to the information seeking process and satisfaction of a user. Since then, defining and inducing serendipity in recommender systems have been important areas of research. Research that examines the potential for serendipity [3] show that there exists ample potential for serendipity in recommender systems and relevance and user behaviour are important factors supporting it. Their algorithms generate novel recommendations based on profiles of user preference, known users and unknown items. Methods for improving user satisfaction by generating unexpected recommendations based on the utility theory of economics was proposed by Adamopolous [1]. Hijikata *et al.* have also proposed a discovery-oriented CF method [17] for deriving novel recommendations. A methodology based on temporal parameters [28] was introduced by Rana to include novelty and serendipity in recommender systems. Ziegler *et al.* [37] assume that diversifying recommendation lists improves user satisfaction and propose topic diversification based on intra-list similarity, which diversifies recommendation lists. An approach by Iaquinta *et al.* [18] suggests recommending items that are semantically far from user profiles. Kawamae [20] assumes that users follow earlier users that demonstrated similar preferences.

Studies on the effect of serendipity in recommender systems are less, yet popular. Most of the research is focused on establishing the need and understanding the heuristics involved in mitigating the overspecialization problem through surprise suggestions. Ge *et al.* emphasize the need to evaluate recommender systems beyond accuracy [14]. Vargas and Castells defined evaluation metrics in this area by proposing a framework built on the concepts of choice, discovery and relevance [35]. They measure and analyze the role played by serendipity as an indicator of recommendation quality. Even though serendipity and unexpectedness involve a positive surprise for the user, it is also important to take into account the relevance of the item to the user. This has been taken into consideration in some works [1]. The algorithm we introduce is fundamentally different from those discussed hitherto. It is a simple clustering based algorithm where each user also gets assigned to a *serendipitous* cluster apart from the conventional cluster it belongs to. SC-CF algorithm incorporates serendipity via SPKM || clustering while maintaining relevance to a user. The following section discusses this in detail.

III. Serendipitous Clustering for Collaborative Filtering (SC-CF)

We present an algorithm for effectuating serendipity in recommender systems. This algorithm has two parts and is configured for movie recommender systems. The first part concerns clustering and the second caters to serendipity. The novelty in our algorithm is two-fold. For clustering, we extend the popular parallel seeding technique of K-Means to the spherical setting with Spherical K-Means parallel (SPKM ||) algorithm. For introducing serendipity, we generate *user profiles* containing information about users' preferences of movie genres and cast. The following two sections elucidate the approach and presents the algorithms.

A. SPKM || clustering

For movie recommender systems, each user is represented by a $x \in \mathbb{R}^n$ (where $\mathbb{R}$ is the set of real numbers), a vector of movie ratings. The cosine similarity between these vectors is the distance metric used for assigning them to clusters. In order to predict user ratings using the Pearson's function (3) The cluster a user belongs to acts as the neighborhood taken into consideration while predicting their rating using the Pearson's function (3) in the collaborative filtering algorithm. The state of art clustering algorithms like K-Means, SPKM and SPKM $++$ are discussed briefly in next few paragraphs. This discussion in turn lead to the motivation behind our Algorithm - SPKM ||.

K-Means clustering is a popular algorithm used in CF [13] to cluster similar users. It minimizes the distance between cluster centers and member data points. On projecting these points onto a unit sphere, the sum of squared distances from the cluster centers acts as a natural measure of dispersion and is the motivation behind Spherical K-Means (SPKM) [33]. It has been proved that the SPKM algorithm solves the problem of sparsity in high dimensional data. For details, refer to the work of Dhillon *et al.* [11].

SPKM $++$ is one variant of the random seeding technique used by SPKM for finding the k initial centers. Endo *et al.* proposed this adaptive sampling algorithm (SPKM$++$) which samples k points based on a probability distribution that maximizes the inter-cluster distances [12]. The seeding step of SPKM$++$ runs for k iterations sampling one point at a time. The clustering thus obtained gives an $\mathcal{O}(\log k)$ competitive result with respect to the optimal clustering. Lloyds-type iterations follow this seeding step in SPKM$++$ and in turn further improves the clustering quality.

However, sampling from such a distribution requires taking k passes over the data, which might be sub-optimal when the values of k, and/or n, dim are large. Here n and dim are the number of user vectors to be clustered and dimensions of those vectors respectively.

The parallel seeding is a technique introduced by Bahmani *et al.* [5]. This method is largely similar to the ++ seeding technique explained by Arthur *et al.* [4] where the initial centers are chosen based on a probability distribution which maximizes the inter-cluster distances. A careful initialization of centers is extremely crucial for obtaining good clustering results. Our goal is to obtain a faster sampling algorithm which gives clustering results very close to that of SPKM$++$. We extend parallel seeding technique to the spherical setting by using cosine similarity which exploits the sparsity of data.

We also present an experimental evaluation on public datasets and show that we are able to achieve a significant speed-up in seeding time with respect to SPKM$++$ for the problem while retaining almost the same accuracy. This clustering method is novel for CF and our experimental validations shown in Section IV demonstrate the superiority of this algorithm in identifying the optimal neighborhood. The pseudocode can be seen in Algorithm 1.

SPKM || requires only $\mathcal{O}(\log n)$ number of passes through the data where n is the number of data points to generate initial centers. In our experiments, the number of passes is denoted by r. In each iteration, l centers are chosen as opposed to one center in ++ seeding [4] (lines 4 to 8 in Algorithm 1). The centers are chosen based on the probability distribution given in line 5 of Algorithm 1. This distribution ensures that the chosen centers are sufficiently far in order to guarantee lesser number of mean updates to obtain optimal clustering. These clusters are called conventional clusters in the context of the SC-CF algorithm.

The major difference between SPKM || algorithm and Scalable K-Means [5] lies in the distance function $d(x, \mathbf{C}_i)$ where $\mathbf{C}_i \in \mathbf{C}$ (list of k cluster centers after seeding). In the spherical setting, this function calculates the cosine similarity between a user x and cluster centers as opposed to Euclidean distances in Scalable K-Means.

$$cosine\_sim(\mathbf{a}, \mathbf{b}) = \langle \mathbf{a}, \mathbf{b} \rangle = \frac{\mathbf{a}.\mathbf{b}}{|\mathbf{a}|.|\mathbf{b}|} \tag{1}$$

Algorithm 1 SPKM || (k, l) seeding

1: $U_x \leftarrow$ sample a user uniformly at random from X {X is the set of all user data points}
2: $n \leftarrow$ number of users
3: $i \leftarrow 0$
4: **while** $i \leq \mathcal{O}(\log n)$ **do**
5: $L \leftarrow$ sample l points from $x \in X$ each with probability $p_x = \frac{l.d(x,U_x)}{J(U_x)}$ {Refer (2)}
6: $U_x \leftarrow U_x \cup L$
7: $i \leftarrow i + 1$
8: **end while**
9: For $\forall\ x \in U_x$, set w_x to be the number of points in X closer to x than any other point in U_x
10: $\mathbf{C} \leftarrow$ sample a single $\mathbf{c} \in U_x$ with probability $w_{\mathbf{x}} / \sum_{\mathbf{c}' \in U_x} w_{\mathbf{c}'}$
11: **for** $i = 2, \ldots k$ **do**
12: Sample $\mathbf{c} \in U_x$ with probability $\frac{w_{\mathbf{x}}(\mathbf{c},\mathbf{C})}{\sum_{\mathbf{c}' \in U_x} w_{\mathbf{c}'}(\mathbf{c}',\mathbf{C})}$
13: $\mathbf{C} \leftarrow \mathbf{C} \cup \{\mathbf{c}\}$
14: **end for**
15: **return** $\mathbf{C}$

For two unit vectors $\mathbf{a}$ and $\mathbf{b}$, the angular dissimilarity, i.e distance between them, is defined as $d(\mathbf{a}, \mathbf{b}) = \gamma - \langle \mathbf{a}, \mathbf{b} \rangle$, where γ is set to be $\geq 3/2$ as theoretically proven by Endo *et al.* [12]. Using this, we define the SPKM clustering objective function as follows:

$$J(\mathbf{C}) = \sum_{i=1}^{k} \sum_{\mathbf{x} \in \pi_i} d(\mathbf{x}, \mathbf{C}_i) = \gamma |\mathcal{X}| - \sum_{i=1}^{k} \sum_{\mathbf{x} \in \pi_i} \langle \mathbf{x}, \mathbf{C}_i \rangle. \quad (2)$$

$J(\mathbf{C})$ represents the clustering cost which in turn reflects the clustering quality. It is calculated as the sum of the distances between each data point and its closest cluster center. Here π_i is the set of points belonging to the i^{th} cluster. As seen from Algorithm 1, the number of users chosen in each iteration is l and we end up with more than k points at the end of this seeding technique. For this reason, each of the points is assigned a weight based on the number of users closest to it (line 9 in Algorithm 1). These $(l \times r) + 1$ points in U_x are then re-grouped into k clusters ($\mathbf{C}$) using weighted K-Means (lines 10 to 14 in Algorithm 1).

B. Serendipitous clustering

The k user clusters ($\mathbf{C}$) obtained after seeding from Algorithm 1 are used to run Lloyd's iterations which in turn returns the final set of k clusters ($\mathbb{C}$). This set is passed as argument to the SC-CF algorithm which follows the work flow shown in Figure 1. In order to effectuate serendipity, the SC-CF algorithm assigns each user to *serendipitous clusters*. These serendipitous clusters are chosen by taking into consideration, the user's profile. A user profile contains two sub-profiles– one with respect to the users' genre preferences and the other corresponding to their cast-director (cast-dir) preferences. A genre profile (*genre_profile*) captures the genre information

Algorithm 2 SC-CF ($\mathbb{C}$, X) for finding serendipitous clusters

1: **for** $\mathbb{C}_i \in \mathbb{C}$ **do**
2: $M_i \leftarrow \forall_{x \in X}\ \text{argmin}_x[\gamma - \langle x, \mathbb{C}_i \rangle]$
3: $\mathbf{M} \leftarrow \mathbf{M} \cup M_i$
4: **end for**
5: **for** $x \in X$ **do**
6: $x_c \leftarrow \mathbb{C}_x$ {the cluster x belongs to}
7: $p \leftarrow \alpha * |x_c|$
8: $q \leftarrow \beta * p$
9: s $\leftarrow$ sorted list on $\langle x$.genre_profile, m.genre_profile$\rangle$ $\forall$ m $\in \mathbf{M}$
10: genre_list $\leftarrow$ top p from list s
11: $l \leftarrow$ sorted list on $J\_sim(x$.cast_profile, g.cast_profile) $\forall$ g $\in$ genre_list {Refer (6)}
12: $S_x \leftarrow$ bottom q from list l
13: $\mathbf{S} \leftarrow \mathbf{S} \cup S_x$
14: **end for**
15: **return** $\mathbf{S}$

of all movies rated by a user. Each movie is represented by a vector where each element is an indicator of whether the movie belongs to a particular genre or not. We consider the most popular 26 genres from the Serendipity 2018 dataset. A cast-dir profile (*cast_profile*) concatenates the list of directors and actors of all the movies a user has rated.

While finding serendipitous clusters for a user, the SC-CF algorithm looks for user genre profiles that are dissimilar from the target user's. This gives us the list of users who have watched movies of genres relatively different from the target user. By doing this, SC-CF captures diversity and helps broaden recommendations.

Following this, SC-CF identifies those clusters whose cast preferences are more similar to that of the target user. In order to identify these clusters, it compares the cast preferences of the target user with that of a "concept" user. In the context of clustering, the data point closest to a cluster center captures and represents the concept of the cluster as a whole. Each user cluster has a concept user denoted by M_i. The same approach is followed while comparing genre preferences. In Algorithm 2, lines 1 to 4 are concerned with finding the concept (M_i) of each cluster. By considering the similarity of cast preferences between users, relevance is maintained. For example, users might like to watch movies starring their favorite actors even if the genre might be less preferred by them.

$$r_{uj} = \mu_u + \frac{\sum_{v \in N} \langle u, v \rangle . (r_{vj} - \mu_v)}{\sum_{v \in N} |\langle u, v \rangle|} \quad (3)$$

Algorithm 2 gives weighted considerations to each of the two sub-profiles. It chooses top p concept users based on dissimilarity of genre profiles and from this list, the top q concept users based on similarity of cast-dir profiles (lines 7 to 12 in Algorithm 2). α is the weight of genre profile dissimilarity and β determines the importance of cast profile similarity taken into consideration. In order to preserve relevance and disallow

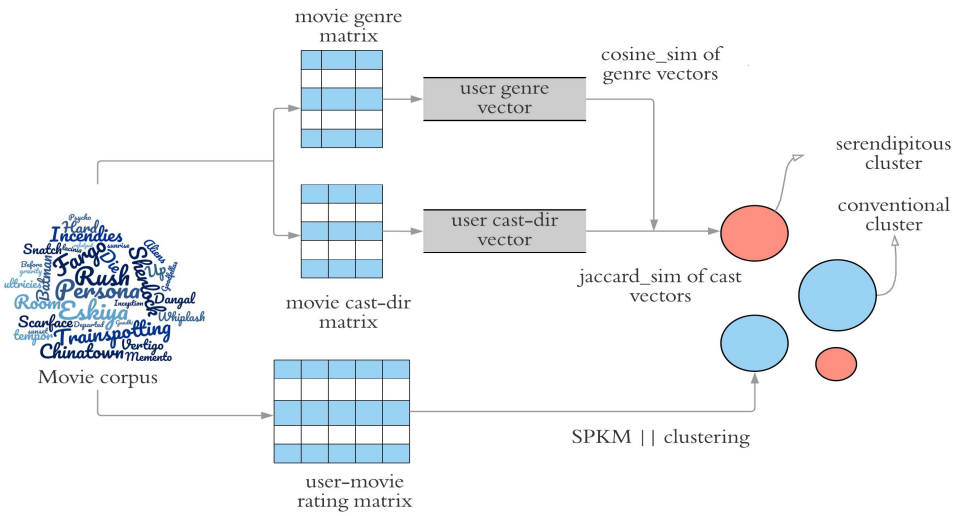

Fig. 1. System workflow of the SC-CF algorithm for identifying serendipitous clusters

TABLE I
REAL-WORLD DATASET DESCRIPTION.

Data Set	# documents	# words in the vocab (dimension)	max # words in a document (sparsity)
KOS blogs	3430	6906	457
ENRON emails	7000	28102	2021

serendipitous items from dominating the list of recommended items for a user, p and q are chosen to be dependent on the size of the conventional cluster that user belongs to. The cosine similarity metric is used to find the conventional cluster a user belongs to and the above described weighted combination is used by SC-CF for identifying the serendipitous clusters of a user.

Once both the conventional and the serendipitous cluster for each user are identified, the prediction function as defined by Shardanand *et al.* [30] is used to calculate the user's predicted rating for a movie. This function as defined by (3) takes the over-all neighborhood of a user into consideration. In normal CF algorithms, the neighborhood is chosen from the conventional cluster that the target user u belongs to. In SC-CF, this neighborhood will also include the serendipitous cluster of u along with the conventional cluster.

In this equation, r_{uj} is the predicted rating value of user u for the movie j. μ_u is the average of all ratings of the user u and N is the neighborhood of users taken into consideration. Recommendations for the user u are made on the basis of these predicted rating values. If the values exceed a threshold, then the corresponding movies are recommended to the user.

IV. Experimental Analysis

A. Dataset

In order to validate SPKM || algorithm, we use two publicly available datasets KOS [39] and ENRON [40]. These dataset were chosen from the UCI Bag of Words repository* and are commonly used for clustering problems. We use all 3430 data points from KOS and only 7000 data points from ENRON. KOS and ENRON are sparse datasets and are available in a format similar to Serendipity 2018. This makes them an apt choice for validating our clustering algorithm. The experiments for end to end SC-CF algorithm were performed on a real world dataset, Serendipity 2018 [21]. Here, we have considered 5,000 users and 49,000 movies from this dataset. This selection of data points is due to computational limitations.

The SC-CF algorithm was evaluated on the Serendipity 2018 dataset released by the MovieLens research group [21]. The dataset includes answers from users about how serendipitous particular movies were to them. It also includes the past ratings of these users and the recommendations they received before answering the questions. Thus, the ratings given by these users to the recommended movies act as good ground-truth measures for calculating accuracy metrics like Root Mean Square Error (RMSE) and Mean Absolute Error (MAE).

B. Methodology

a) Evaluating SPKM ||: To demonstrate the superior performance of the SPKM || clustering algorithm, we compare its performance with the state-of-the-art clustering algorithms SPKM [11] and SPKM++ [12]. Here we do not compare with

*https://archive.ics.uci.edu/ml/datasets/bag+of+words

TABLE II
COMPARISON OF CLUSTERING QUALITY BETWEEN SPKM AND SPKM++ FOR DIFFERENT VALUES OF k ON THE KOS DATASET.

k	SPKM (Clustering cost)	SPKM++ (Seeding cost)	SPKM (Total running time(s))	SPKM++ (Seeding time(s))
3	3974.64	3958.73	159.17	9.64
5	3832.52	3832.57	369.67	31.65
7	3788.65	3781.33	531.34	64.66
10	3747.02	3736.83	616.64	139.93

TABLE III
ENTRIES OF THE TABLE DESCRIBE THE RELATIVE IMPROVEMENT IN THE CLUSTERING COST OF SPKM || WITH RESPECT TO THE COST OF SPKM++ ON THE KOS DATASET.

k	SPKM++	SPKM || (Clustering cost)		
	(Clustering cost)	l = k/2	l = k	l = 2k
20	0.00%	0.06%	-0.04%	0.06%
50	0.00%	0.15%	0.22%	0.07%
100	0.00%	0.19%	-0.002%	0.1%
150	0.00%	0.02%	0.04%	-0.03%
200	0.00%	0.08%	0.11%	-0.18%

TABLE IV
ENTRIES OF THE TABLE DESCRIBE THE RELATIVE IMPROVEMENT IN THE CLUSTERING COST OF SPKM || WITH RESPECT TO THE COST OF SPKM++ ON THE ENRON DATASET.

k	SPKM++	SPKM || (Clustering cost)		
	(Clustering cost)	l = k/2	l = k	l = 2k
20	0.00%	3.33%	-6.79%	3.3%
50	0.00%	0.04%	0.08%	-0.12%
100	0.00%	0.11%	0.19%	0.08%
150	0.00%	0.06%	0.09%	-0.03%
200	0.00%	0.12%	0.08%	0.04%

the classical K-Means clustering [9] because the underlying objective that is optimized in that algorithm is different and hence such a comparison would be incorrect. In K-Means, the goal is to cluster points represented in a Euclidean space such that the sum of squared distances from each point to its nearest cluster center is minimized. On the other hand in SPKM ||, the goal is to cluster points represented on a unit sphere by maximizing the sum of the cosine similarity from each point to its nearest cluster center.

First we compare the clustering cost between SPKM and SPKM++ for different values of k to establish that the latter produces better quality clusters as evidenced by the lower clustering cost $J(\mathbf{C})$ (2). We run this experiment on both the KOS and the ENRON datasets but display the results only for the KOS (Table II). Similar results were obtained on the ENRON dataset. Ideal clustering requires very low intra-cluster distances. Since the clustering cost is the sum over distances of points to their closest centers, a lower clustering cost indicates higher clustering quality.

The SPKM || algorithm is then compared in terms of clustering cost (Tables III and IV) and seeding time (Figures 2 and 3) with the SPKM++ algorithm [4] on both the KOS and the ENRON datasets.

b) Identifying best values for parameters k and l: The SPKM || part of the SC-CF algorithm is dependent on the values of two parameters – k, the number of clusters and l, the number of points sampled in each iteration of the seeding. It is important to identify the values of these parameters for which the algorithm produces best results. These results can be compared using the RMSE and MAE scores as defined below.

$$RMSE = \sqrt{\frac{1}{T}\sum_{i,j}(r_{ij} - \hat{r}_{ij})^2} \tag{4}$$

$$MAE = \frac{1}{T}\sum_{i,j}|r_{ij} - \hat{r}_{ij}|^2 \tag{5}$$

T here is the number of test ratings, r_{ij} is the rating given to item j by user i and $\hat{r}_{ij}$ is the predicted rating value by SC-CF. Smaller RMSE and MAE values indicate better recommendation results.

c) Comparing SC-CF with baseline techniques: The SC-CF algorithm is evaluated on the Serendipity 2018 dataset. The RMSE and MAE values obtained are compared with those of both K-Means and deep learning based [7], [36] state-of-art techniques.

d) SC-CF measured in terms of unexpectedness and diversity: Evaluating the serendipitous recommendation algorithm using previously construed metrics [19], [37] of unexpectedness and diversity may not be ideal as these are dependent on the definitions of these terms. Hence, we define metrics for measuring the extent of diversity and unexpectedness produced in the recommendations by our algorithm. We measure unexpectedness by comparing the list of recommendations produced by the SC-CF algorithm and the list of recommendations produced without considering the serendipitous clusters. This comparison is captured by the Jaccard similarity [15] metric and is averaged over all users.

$$J\_sim(L_u, S_u) = \frac{|L_u \cap S_u|}{|L_u \cup S_u|} \tag{6}$$

The unexpectedness of our recommended list is given by the following equation.

$$Unexpectedness = \frac{1}{|U|}\sum_{u} 1 - J\_sim(L_u, S_u) \tag{7}$$

Here, U is the set of target users for whom recommendations are generated. L_u is the list of movies recommended

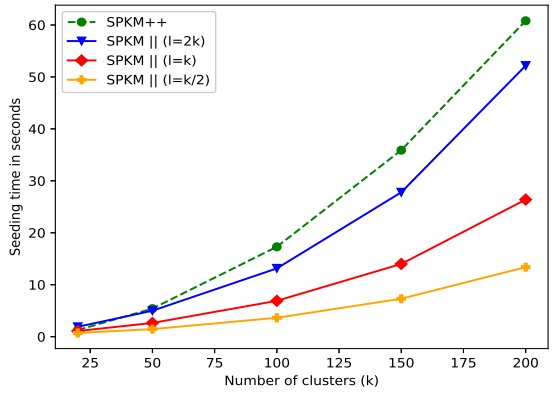

Fig. 2. The graph shows the comparison between the seeding time of SPKM || and popularly used SPKM++ algorithms on the KOS dataset.

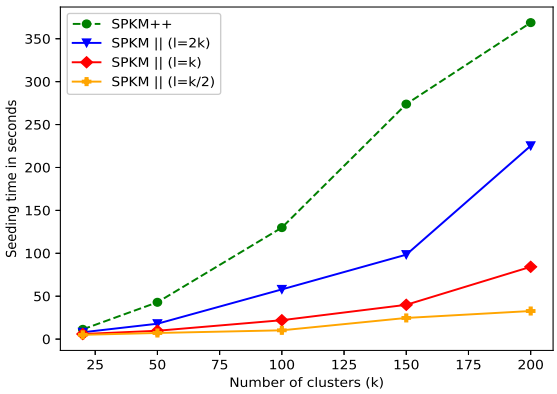

Fig. 3. The graph shows the comparison between the seeding time of SPKM || and popularly used SPKM++ algorithms on the ENRON dataset.

for user u without considering serendipitous clusters and S_u is the list of movies recommended to user u by the SC-CF algorithm.

$$Diversity = \frac{1}{|U|}\sum_{u}\frac{1}{|S_u|}\sum_{i\in S_u}\sum_{j\in S_{u,j\neq i}} Hamm(i,j) \quad (8)$$

In order to measure the diversity present in the list of recommended movies generated by SC-CF, we look at the pairwise distances between movies in S_u. These distances are computed as the normalized Hamming distance ($Hamm$) between the genre profiles of the movies present in S_u. The reason we consider only genre profile is because users were assigned to serendipitous clusters based on the dissimilarity of genre and similarity of cast. Hence, the diversity is captured by the genre profiles alone. We calculate the diversity value by averaging the normalized Hamming distance for all users. The equation (8) can be used to calculate the diversity.

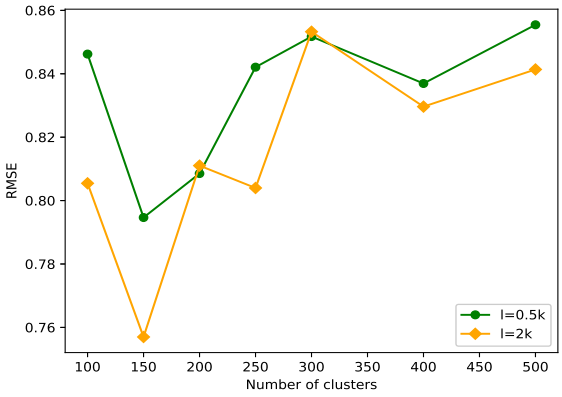

Fig. 4. Empirically deciding the neighborhood size from RMSE of recommendations

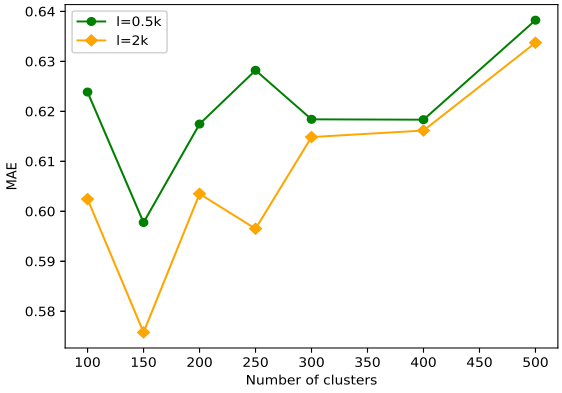

Fig. 5. Empirically deciding the neighborhood size from MAE of recommendations

V. Results and Discussion

This section demonstrates the recommendation performance of both SPKM || and the SC-CF algorithm by comparing them with state-of-art techniques on real world datasets.

All the clustering-based experiments discussed below were repeated three times in order to reduce the effect of randomness and the average results are displayed. For all SPKM methods, the Lloyd's iterations are terminated when there is no change in the set of chosen cluster centers. All experiments were run on a 2.3 GHz Intel Xeon machine (AWS instance of type m4.2xlarge) with 32GB memory.

A. Evaluating SPKM ||

To evaluate the clustering cost of SPKM ||, we compare it against approaches involving different initialization techniques. The initialization methods are implicitly followed by Lloyd's iterations [16].

In our first set of experiments, we compare the clustering quality produced by SPKM and SPKM++ on the KOS dataset

for $k = \{3, 5, 7, 10\}$ and the results can be viewed in Table II. The second column of the table shows that SPKM++ algorithm gives better clustering quality than SPKM right after seeding itself and this is also obtained in lesser time (see columns 3 and 4). Clustering results of SPKM++ are subject to further improvement after Lloyds iterations on the results of the seeding step. Thus we verify the established fact that SPKM++ algorithm produces better clusters than SPKM. Hence all further experiments compare the SPKM || algorithm with SPKM++ alone.

In the next set of experiments, we compare the SPKM || algorithm with SPKM ++ on KOS and ENRON for k values ranging from 20 to 200 for $l = \{0.5k, k, 2k\}$ with $r = 4$ rounds for all the cases. Tables III and IV show that the SPKM || maintains similar clustering cost as SPKM ++. (These chosen values for l are consistent with Bahmani *et al.* [5]).

The running time of SPKM || consists of two components: the time required to identify the initial cluster points and the running time of Lloyd's iterations till convergence. The former is proportional to both the number of passes through the data and to the size of the intermediate solution. The after-seeding part in both SPKM || and SPKM++ (Lloyd's iterations) is the same and hence there is no need for comparing them. From Figures 2 and 3 it is clear that the running time of the initialization routine in SPKM || is much smaller than that by SPKM ++. The seeding time in SPKM++ increases with the value of k whereas this increase is much slower for SPKM || as can be seen from Figures 2 and 3.

In the KOS dataset, the speed-up (when $k = 200$) is $4.9\times$ for $l = 0.5k$, $2.6\times$ for $l = k$ and $1.3\times$ for $l = 2k$. In the ENRON dataset which is larger than KOS, the speedup is significantly higher as can be observed from Figure 3. The speedup (when $k = 200$) is $11.3\times$ for $l = 0.5k$, $4.4\times$ for $l = k$ and $1.6\times$ for $l = 2k$. Intuitively, a lower value of l results in faster seeding as the number of chosen cluster centers is smaller.

B. The neighborhood selection by identifying best values of k and l

The problem of neighborhood selection is of prime importance and algorithms must consider only those users who have a positive correlation with the target user. The common techniques for reducing the size of the neighborhood include defining a specific minimum threshold of user similarity, or limiting the size to a fixed number and taking only some nearest neighbors into account. The potential problems of both techniques are discussed by Anand and Mobasher [26]. If the similarity threshold is too high, the size of the neighborhood is very small for many users which reduces the coverage. When the number of neighbors taken into account is too high, the 'extra' users add additional noise which affects the RMSE value. Clustering approaches provide a solution for this by facilitating the automatic selection of neighborhood size. The K-Means algorithm and its variants try to minimize the sum squared error between the data points and cluster centers where each individual cluster size is constrained by prior knowledge about the data set. In the scenario of recommender systems for movies, this knowledge is the rating data for conventional clustering and genre, cast and director information for serendipitous clustering. Hence, the cluster that a user belongs to acts as a good neighborhood.

In our second set of experiments, we explore the choice of l and k when the sampling is done with replacement, as specified in the SPKM || algorithm. It is important to note that this ideal value of k depends on the number of points in the data corpus and hence varies with the dataset being used. Figures 4 and 5 illustrate the values of k and l that give the best accuracy and all following experiments use these values. The best prediction result was obtained with the combination $k = 150$ and $l = 2k$, yielding an RMSE of 0.757. Different values of l follow the same trend of RMSE values for varying values of k and achieve the best performance with number of clusters $k = 150$.

C. Comparing SC-CF with baselines

In order to investigate the performance of SC-CF recommendations, we compare our algorithm with SVD++ [23], BiasedMF [24], Continuous Restricted Boltzmann Machine (CRBM) [7] and DNN [36]. We also compare the MAE of SC-CF with the EHC-CF [38]. These algorithms have been briefly described in Section II.

Figure 6 shows a comparison of the above techniques with reference to the RMSE and MAE values. It can be observed that SC-CF outperforms all algorithms except DNN. SC-CF ($l = 0.5k$ and $k = 100$) achieves 0.79 RMSE and 0.60 of MAE. All SC-CF variants with $l = 2k$ and $k = \{100, 150, 200, 250\}$ beat SVD++ [23], BiasedMF [24] and CRBM [7]. The best version of SC-CF outperforms these popular existing methods by reduced average prediction error of 9%. However, the RMSE and MAE scores are very close to those achieved by the Vector space embedding+DNN method [36]. The training of DNN models depends on a large number of parameters and other resources and hence is cumbersome for network optimization. On the other hand, our algorithm finds an optimal neighborhood in an effective manner with considerably lower investments while successfully achieving similar RMSE and MAE scores. Also, the clustering approach appears more explainable than neural networks in recommender systems.

In our next set of experiments, the first compared algorithm is user-based collaborative filtering without clustering (CF). The second compared algorithm is user-based collaborative filtering with K-Means clustering method (Kmeans-CF). The third and last one is the EHC-CF algorithm proposed by Chen *et al.* [38]. Figure 7 shows the significant improvement in MAE demonstrated by the SC-CF algorithm over CF, K-Means CF and EHC-CF algorithms. The best MAE values of CF, Kmeans-CF and EHC-CF are 0.878, 0.843 and 0.831 respectively. MAE of EHC-CF is lower than CF which means that collaborative filtering with clustering gains accuracy. The SC-CF with $l = 2k$ and $k = 150$ reduces the average

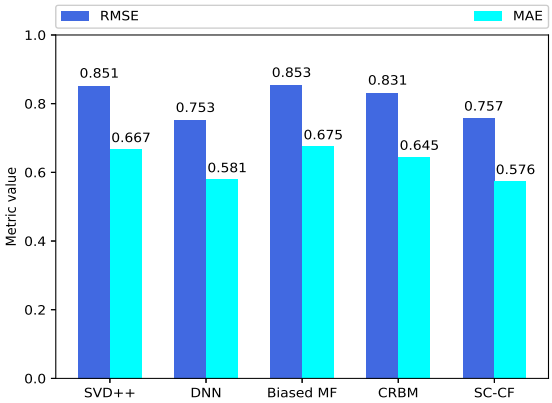

Fig. 6. RMSE and MAE values of baselines compared with SC-CF

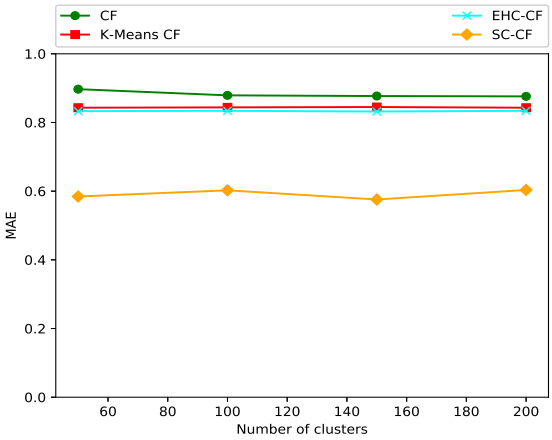

Fig. 7. RMSE and MAE values of clustering-based techniques compared with SC-CF

prediction error by 31% compared to the EHC-CF clustering technique.

We do not compare our results with other serendipitous algorithms because we feel that the lack of a common definition for "serendipity in recommender systems" would make such comparisons meaningless. The metrics used by other algorithms are dependent on the factors used by them to define serendipity and hence using these to evaluate our algorithm may be deemed unfair.

TABLE V
DIVERSITY AND UNEXPECTEDNESS FOR DIFFERENT α AND β

α	β	Diversity	Unexpectedness
0.25	0.50	0.424	0.807
0.25	0.75	0.416	0.843
0.75	0.25	0.430	0.862
0.5	0.25	0.382	0.893

1) SC-CF measured in terms of unexpectedness and diversity: In Table V, we calculate the diversity and unexpectedness for different sets of values of α and β. It can be observed that even with varying values of the parameters, there is very little change in the diversity and unexpectedness values. α decides the weight of genre dissimilarity and β determines the weight of cast-dir similarity. However, our algorithm produces recommendations that are consistent in terms of diversity and unexpectedness thus implying little or no dependence on the values of these parameters.

From our final set of experiments, we observe that SC-CF produces almost 41% diversity which signifies that the list of recommendations generated consists of movies that belong to a diverse range of genres. On an average, 85% of the recommended list generated by SC-CF consists of movies unexpected by a user.

VI. CONCLUSIONS

Through this work, we show that content-based user clustering can be enriched with serendipity in recommendations. Traditionally, collaborative filtering systems have relied largely on the rating profiles of users as a way to build the optimal set of neighbors. In this paper, we have argued that profile similarity on its own may not be sufficient and other factors like diversity and unexpectedness also have an important role to play. This paper reports an effort in applying the global idea like serendipity in clustering techniques for recommender systems. The SC-CF algorithm succeeds in introducing serendipity while maintaining relevance without harming RMSE, and thus breaks with the general notion that serendipitous recommendations harm model accuracy.

The clustering technique (SPKM ||) in SC-CF efficiently addresses the problem of sparsity by giving suitable neighborhoods for target users. SPKM || gives improved clustering quality with lower number of rounds compared to the widely used ++ seeding technique. It is also scalable with increase in number of users and dimensionality of the user-rating vectors. The SC-CF algorithm incorporates the notion of "serendipitous clusters" via the SPKM || clustering. SC-CF was evaluated on a standard dataset against benchmark approaches. We have found the use of serendipitous clusters to have a positive impact on overall prediction error rates, with the best performing strategy reducing the average prediction error by 31% compared to the other clustering techniques [38] in CF. Experiments with recent approaches [24], [23] including deep models [7] showed that SC-CF consistently and significantly outperforms the popular existing methods and reduces the average prediction error by 9%. SC-CF also achieves RMSE and MAE which is on par with the Embedding+DNN model [36] while effectuating serendipity. It is likely that the user's overall liking of recommendation lists goes beyond accuracy and involves other factors, e.g., the user's perceived list diversity. We were able to obtain empirical evidence that recommendations generated by SC-CF account for diversity and unexpectedness, and thus bear an intrinsic added value.

The idea of including serendipitous clusters in a user's neighborhood can be extended to different domains of recommender systems like books, news, travel, etc. In this work we have focused on generating user-profiles which take into account features from the movie preferences. In order to extend this to other domains, choosing simple features relevant in those domains should be sufficient as the underlying clustering algorithm would work in the same way.

REFERENCES

[1] P. Adamopoulos and A. Tuzhilin, On Unexpectedness in Recommender Systems, ACM Transactions on Intelligent Systems and Technology, vol. 5, no. 4, pp. 1–32, 2014.

[2] N. Ampazis, Collaborative filtering via concept decomposition on the netflix dataset, In ECAI, vol. 8, pp. 26–30, 2008.

[3] P. Andr, J. Teevan, and S. T. Dumais, From x-rays to silly putty via Uranus, Proceedings of the 27th international conference on Human factors in computing systems - CHI 09, pp. 2033–2036, 2009.

[4] D. Arthur and S. Vassilvitskii, k-means : The advantages of careful seeding., Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, pp. 1027–1035, 2007.

[5] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, Scalable k-means , Proceedings of the VLDB Endowment, vol. 5, no. 7, pp. 622–633, Jan. 2012.

[6] O. Celma, Music Recommendation, Music Recommendation and Discovery, pp. 43–85, 2009.

[7] H. Chen and A. Murray, Continuous restricted Boltzmann machine with an implementable training algorithm, IEE Proceedings - Vision, Image, and Signal Processing, vol. 150, no. 3, pp. 153–158, 2003.

[8] J. Chen, Uliji, H. Wang, and Z. Yan, Evolutionary heterogeneous clustering for rating prediction based on user collaborative filtering, Swarm and Evolutionary Computation, vol. 38, pp. 35–41, 2018.

[9] G. M. Dakhel and M. Mahdavi, A new collaborative filtering algorithm using K-means clustering and neighbors voting, 2011 11th International Conference on Hybrid Intelligent Systems (HIS), pp. 179–184, 2011.

[10] I. S. Dhillon and D. M. Modha, Large-scale parallel data mining, LectureNotes in Artificial Intelligence 1759, pp. 245–260, 2000.

[11] Dhillon, Inderjit S., and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. Machine learning 42.1-2 (2001): 143-175.

[12] Y. Endo and S. Miyamoto, Spherical k-Means Clustering, Modeling Decisions for Artificial Intelligence Lecture Notes in Computer Science, pp. 103–114, 2015.

[13] N. Ganganath, C.-T. Cheng, and C. K. Tse, Data Clustering with Cluster Size Constraints Using a Modified K-Means Algorithm, 2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, pp. 158–161, 2014.

[14] M. Ge, C. Delgado-Battenfeld, and D. Jannach, Beyond accuracy, Proceedings of the fourth ACM conference on Recommender systems - RecSys 10, pp. 257–260, 2010.

[15] D. W. Goodall, A New Similarity Index Based on Probability, Biometrics, vol. 22, no. 4, pp. 882–907, 1966.

[16] J. A. Hartigan and M. A. Wong, Algorithm AS 136: A K-Means Clustering Algorithm, Applied Statistics, vol. 28, no. 1, pp. 100–108, 1979.

[17] Y. Hijikata, T. Shimizu, and S. Nishida, Discovery-oriented collaborative filtering for improving user satisfaction, Proceedingsc of the 13th international conference on Intelligent user interfaces - IUI 09, pp. 67–76, 2009.

[18] L. Iaquinta, M. De, P. Lops, G. Semeraro, and P. Molino, Can a Recommender System Induce Serendipitous Encounters?, E-commerce, Jan. 2010.

[19] M. Kaminskas and D. Bridge, Diversity, Serendipity, Novelty, and Coverage, ACM Transactions on Interactive Intelligent Systems, vol. 7, no. 1, pp. 1–42, 2017.

[20] N. Kawamae, Serendipitous recommendations via innovators, Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval - SIGIR 10, pp. 218–225, 2010.

[21] D. Kotkov, J. A. Konstan, Q. Zhao, and J. Veijalainen, Investigating serendipity in recommender systems based on real user feedback, Proceedings of the 33rd Annual ACM Symposium on Applied Computing - SAC 18, 2018.

[22] D. Kotkov, J. Veijalainen, and S. Wang, Challenges of Serendipity in Recommender Systems, Proceedings of the 12th International Conference on Web Information Systems and Technologies, ISBN 978-989-758-186-1, vol. 2 , 2016.

[23] R. Kumar, B. K. Verma, and S. S. Rastogi, Social Popularity based SVD Recommender System, International Journal of Computer Applications, vol. 87, no. 14, pp. 33–37, 2014.

[24] X. Liu and W. Wu, Learning Context-aware Latent Representations for Context-aware Collaborative Filtering, Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR 15, 2015.

[25] A. Maksai, F. Garcin, and B. Faltings, Predicting Online Performance of News Recommender Systems Through Richer Evaluation Metrics, Proceedings of the 9th ACM Conference on Recommender Systems - RecSys 15, pp. 179–186, 2015.

[26] Intelligent Techniques for Web Personalization, IJCAI 2003 Workshop, ITWP 2003Acapulco, Mexico, August11, 2003, Revised Selected Papers. Vol. 3169. Springer.

[27] A. Narang, R. Gupta, A. Joshi, and V. K. Garg, Highly scalable parallel collaborative filtering algorithm, 2010 International Conference on High Performance Computing, pp. 1–10, 2010.

[28] C. Rana, New dimensions of temporal serendipity and temporal novelty in recommender system, Advances in Applied Science Research 4, 1 (2013), pp. 151–157, 2013.

[29] P. Resnick and H. R. Varian, Recommender systems, Communications of the ACM, vol. 40, no. 3, pp. 56–58, Jan. 1997.

[30] U. Shardanand and P. Maes, Social information filtering, Proceedings of the SIGCHI conference on Human factors in computing systems - CHI 95, pp. 210–217, 1995.

[31] A. Spink, H. Greisdorf, and J. Bateman, From highly relevant to not relevant: Examining different regions of relevance1, Information Processing & Management, vol. 34, pp. 599–621, 1998.

[32] S. Suboojitha, Introducing serendipity in recommender systems through collaborative methods, 2014.

[33] Hill, Mark O., Colin A. Harrower, and Christopher D. Preston. "Spherical kmeans clustering is good for interpreting multivariate species occurrence data." Methods in Ecology and Evolution 4.6 (2013): 542-551.

[34] E. Tacchin, Serendipitous mentorship in music recommender systems, 2012.

[35] S. Vargas and P. Castells, Rank and relevance in novelty and diversity metrics for recommender systems, Proceedings of the fifth ACM conference on Recommender systems - RecSys 11, 2011.

[36] Y. Wang and P. Craig, Vector space model embedding for recomender system neural networks, 2017 4th International Conference on Systems and Informatics (ICSAI), 2017.

[37] C. N. Ziegler, S. M. Mcnee, J. A. Konstan, and G. Lausen, Improving recommendation lists through topic diversification, Proceedings of the 14th international conference on World Wide Web - WWW 05, 2005.

[38] J. Chen, H. Wang, and Z. Yan, Evolutionary heterogeneous clustering for rating prediction based on user collaborative filtering, Swarm and Evolutionary Computation, vol. 38, pp. 35–41, 2018.

[39] M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml

[40] Klimt, Bryan, and Yiming Yang. The enron corpus: A new dataset for email classification research. European Conference on Machine Learning. Springer, Berlin, Heidelberg, 2004.

[41] Arthur, David, and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 2007.