Novel Centroid Selection Approaches for KMeans-Clustering Based Recommender Systems

Sobia Zahra, Mustansar Ali Ghazanfar, Asra Khalid, Muhammad Awais Azam, Usman Naeem*, Adam Prugel-Bennett**

Department of Software Engineering, Faculty of Telecommunication and Information Engineering, University of Engineering and Technology, Taxila, Pakistan,
*Email: zahra.sobia@gmail.com, mustansar.ali@uettaxila.edu.pk, asra_05@yahoo.com, awais.azam@uettaxila.edu.pk, *U.Naeem@uel.ac.uk,**apb@ecs.soton.ac.uk;*
Phone: +92 (051) 9047 566; fax: +92 (051) 9047 420
**School of Architecture, Computing and Engineering, University of East London, Docklands, London, United Kingdom*
***School of Electronics and Computer Science, University of Southmapton, SO17 1BJ, Southampton, United Kingdom*

Abstract

Recommender systems have the ability to filter unseen information for predicting whether a particular user would prefer a given item when making a choice. Over the years, this process has been dependent on robust applications of data mining and machine learning techniques, which are known to have scalability issues when being applied for recommender systems. In this paper, we propose a k-means clustering-based recommendation algorithm, which addresses the scalability issues associated with traditional recommender systems. An issue with traditional k-means clustering algorithms is that they choose the initial k centroid randomly, which leads to inaccurate recommendations and increased cost for offline training of clusters. The work in this paper highlights how centroid selection in k-means based recommender systems can improve performance as well as being cost saving. The proposed centroid selection method has the ability to exploit underlying data correlation structures, which has been proven to exhibit superior accuracy and performance in comparison to the traditional centroid selection strategies, which choose centroids randomly. The proposed approach has been validated with an extensive set of experiments based on five different datasets (from movies, books, and music domain). These experiments prove that the proposed approach provides a better quality cluster and converges quicker than existing approaches, which in turn improves accuracy of the recommendation provided.

Keywords: Recommender systems, Collaborative filtering, K-means clustering, Centroid (seed) selection in k-means clustering

1. Introduction

1.1. Recommender systems

The need for recommendations from trusted sources is triggered when it is not possible to make choices with insufficient personal experience of a particular domain. This is also seen as a natural phenomenon of the human decision making process [39]. In today's digital era, we are overwhelmed with volumes of information, where processing all of this information is beyond human capabilities. This proliferation of available digital information (i.e. music in LastFm (last.fm), videos in Netflix (netflix.com) and YouTube (youtube.com), electronic resources (i.e. research papers in CiteULike (citeulike.org)), and on-line services (i.e. Amazon (amazon.com), Delicious (delicious.com), Flicker (flickr.com))) need an automated tool to extract and present preferred information to the user. This information also needs to be prioritized, as management and usage of this data can be over-whelming.

Although search engines provide the most relevant pages in response to a user query, however it is still a challenge to generate specific recommendations based on a series of simple keywords. Hence there is a need for information filtering system, which extracts the most relevant unseen information based on the assumption that the user will like the resource. These systems are called recommender systems, which fulfill aforementioned needs and guide users to choose the best resource among all the available resources. Various algorithms are used to model peoples' preferences, predict rating of the resource, and provide recommendation. BusinessInsider [35] reports Google's news algorithms, Facebook new feeds, Netflix and Amazon.com recommendation engines are among the 11 most essential algorithms that make the Internet works. Recommender systems are also being used in an academic setting to provide the relevant research resources [48, 55, 58].

1.2. Formulation of Recommender systems

Item and user both are finite sets, We denote all these users by $\mathcal{U} = \{u_1, u_2, \cdots, u_M\}$ and items by $\mathcal{I} = \{\ i_1, i_2, \cdots, i_N\ \}$. While $|\mathcal{U}| = M$ denotes the total

number of users and $|\mathcal{I}| = N$, denotes the total number of items. Rating is numerical representation of user's interest in any item. This quantitative measure facilitates the recommender systems to exploit user's preferences for providing better recommendation. The rating set by the whole community is mostly represented in the form of a matrix, called User-Item matrix. The rows of matrix are the users in the community and columns displays the items in the system. The term $r_{i,u}$ denotes a matrix entry i.e. rating provided by user u to item i. These ratings are denoted by $(r_{i,u}|(i,u) \in \mathcal{D})$. A single user usually rates small number of items, where set $\mathcal{D} \subset \mathcal{I} \times \mathcal{U}$ refers to users-item pairs that have been rated. A rating matrix R of dimensions $M \times N$ stores the set of possible ratings made by the user. All ratings by user u is denoted by $\mathcal{D}_u$ while the set of users who have rated an item i by $\mathcal{D}_i$. The aim of recommender system is to predict an unseen rating $r_{i,u}$, i.e. for $(i,u) \notin \mathcal{D}$.

1.3. Main Types of Recommender Systems

There are two main types of recommender systems—Collaborative Filtering (CF) and Content-Based Filtering (CBF) recommender systems—as follows:

1. *Collaborative Filtering (CF)*: Collaborative filtering is considered to be the most popular approach for recommendation systems [17]. The collaborating filtering takes into account the interests of similar users, under the assumption that the active users will be interested in items that users similar to them have rated highly. This technique is used by Amazon (amazon.com), iTunes, GroupLens system [32] and Ringo (www.ringo.com). Collaborative filtering can be classified into two sub-categories as follows:
   - *Memory-based approaches* make predictions by taking into account the active user's rating data. All the ratings provided by the users are kept in memory and used for predictions. To compute similarity between items/users all the previously rated items are considered. This approach is used by GroupLens recommender systems [32].
   - *Model-based approaches* initially train a model based on training data and then makes prediction for real data. Usually these models are based on clustering or classification techniques and are used to find pattern from training set. Typical examples are clustering models [4, 24, 44], Kernel-mapping recommender [23, 25, 26], and Singular Value Decomposition (SVD) based models [22, 63].
2. *Content-Based filtering*: Content-based filtering approach uses textual (content) features of items in order to make recommendations. These approaches train machine learning classifiers over user's and item's profiles. An example of content-based systems is Pandora (www.pandora.com) that uses the attributes (properties) of a song (or artist) a user is currently playing (or listening to) in order to generate recommendations of music with similar properties. Some other well-known approaches include [42, 43].

Furthermore, hybrid recommender systems have been proposed [7, 10, 13, 18, 21, 36, 38], which combine individual recommender systems to avoid certain limitations of individual recommender systems.

Recommendations can be presented to an active user in two different ways:

1. Predicting ratings of item that the user has not seen.
2. Construct a list of items ordered by the users preferences, which is known as top-N recommendations.

In this paper, we will investigate both of these approaches.

1.4. Problem Statement and Design Objectives

K-Means-based CF recommendation approaches have been proposed to solve the recommendation problem [7, 49, 54, 64, 66, 67]; however, using random seeds prior to cluster the user-item rating matrix, which has been heavily used in the literature, is not a reasonable approach.

The proposed work aims to improve the quality of clusters and recommendations by investigating different centroid selection approaches and analysing how they affect the quality of clusters/recommendations. We have proposed various centroid selection algorithms using domain specific characteristics, with the following design objectives:

- *Accuracy*: The accuracy is one of the most important design objectives in recommender system's community. The reason being, if a customer leverages a recommender system and then discover that they are getting false recommendations, it is unlikely they will continue using the system. Consequently, an algorithm should make accurate recommendations. We want a recommendation algorithm to be accurate than the conventional K-Means based CF algorithms.
- *Scalability*: A recommendation algorithm should scale gracefully with the increase in data. The conventional collaborative filtering based methods fail to achieve good scalability. The conventional K-Means clustering algorithms cluster the users into different groups; however, choosing random initial centroids can lead to slow convergence. It must be noted that a recommender system's robustness is measured by two factors—accuracy and scalability. These factors are in conflict, as the less time an algorithm takes building the clusters, the more scalable it will be; however it might lead to poor quality recommendations due to partially formed clusters.
- *Cluster quality*: The idea of clustering algorithm is to separate users into different groups based on their similarity. But as clustering is very computationally

difficult task (NP-hard problem) [4]; hence, many local solutions are possible. Since the conventional K-Means algorithms choose initial centroids randomly hence they can converge to local optima [4] resulting in poor quality clusters (refer to Section 5.2 for cluster quality).

- *Coverage*: The coverage (the number of test samples an algorithm can make prediction for [25]) of a recommendation algorithm should be maximum. However, as the K-Means CF-based recommendation approach uses random seeds to cluster the data; hence, the quality of the clusters might suffer with these seeds that might degrades recommendation coverage.

- *Robustness to sparsity*: The performance of a recommendation algorithm should not degrade sharply under sparse datasets (refer to Section 4.1 for sparsity). Figure B.10 shows how different centroid selection approaches work under sparse condition. Result shows that convention k-means suffer the most under sparsity.

- *Cold start problem*: Usually for testing recommender systems, some ratings from the dataset are used for training purpose while other are treated as unseen ratings, which are used to test the performance of proposed solution. Mostly the dataset is selected having users with large number of ratings, in order to achieve accurate recommendations. But to make practical recommendation on real application, we come across different issues. Real application generally have highly skewed data, that is a large number of items may have received just few ratings and a large number of users may have just provided very small number of rating. So it become really tough provide reliable recommendation which can attract new users. CF-based algorithms often come across two important cold-start problem as given below:

  - *New user cold-start problem*: As the CF recommender system generates recommendations based on liking and disliking of active user. When a new user enters the system, the system don't have much information about the user, also the user is not aware of product/services and he/she is hesitant to rate them, hence reliable recommendation can not be provided to that user. This is called new user cold-start problem [1]. We have proposed solution for this problem and the results are shown in Table 10 and Table 11.
  - *New item cold-start problem*: As CF recommender system relies on users' rating, so the item is recommended to the user based on the ratings provided by other users to that item. Initially when a new item is introduced in the system, the system don't have any rating for that item and an item can not be recommended to any user until it gets significant ratings. In CF approach it is tough to get ratings for new item from significant number of users. This is called new item cold-start problem [1]. We have proposed solution for this problem and the results are shown in Table 12 and Table 13.

- *Long tail problem*: : In real application all the items in the dataset are not rated by significant number of users. Some of the unpopular or newly introduced items may have relatively small number of ratings. CF-based recommender systems are unable to provide reliable recommendations for such items or sometimes simply ignore them. This problem is known as long tail problem and most of the items in recommender systems fall in this category [46]. As these item can't be left overlooked and there is a need to develop some algorithm that can filter and provide accurately recommendations from the items that exist in long tail category. Our proposed approaches solve this problem and the results are shown in Table 14 and Table 15.

Against the aforementioned research objectives, this paper aims at developing new K-Means-based recommendation approaches and comparing them with other K-Means CF-based recommendation approaches proposed in literature.

The rest of the paper is organised as follows. In Section 2, we present the related work by giving an overview of different clustering algorithms that have been used for recommendation purposes. In Section 3, we present various centroid selection algorithms. We briefly describe the experimental setup in Section 4. In Section 5, we present the results in detail followed by the conclusion in Section 6.

2. Related Work

2.1. Clustering in Recommender Systems

Clustering is an unsupervised classification approach for recognising patterns, which is based on grouping similar objects together. This approch is useful for finding patterns in an unlabeled dataset. Machine learning, bioinformatics, image analysis, pattern recognition and outlier detection are few of many application areas of clustering [2]. Two major approaches of clustering are hierarchical and partitional clustering.

- **Hierarchical Clustering**: Hierarchical clustering [3] produces nested series of partitions whether it is agglomerative or divisive. In agglomerative approach every pattern is placed in distinct cluster and clusters keep merging based on similarity until any desired condition is met. While in divisive method inverse happens, all the patterns are placed in single cluster

and clusters keep splitting until any stopping criterion is met. Hierarchical clustering produces a dendogram showing pattern and different similarity levels of grouping. Single linkage and complete linkage are most popular examples of hierarchical clustering while some variants of these are also used.

- **Partitional Clustering**: Partitional clustering [29] aims at finding the single partition rather than numerous as in hierarchical methods. Its benefit is that it can be applied on large datasets for which dendogram does not work. But its quality depends upon choice of number of output clusters. It can be applied on whole dataset in order to find out global optimal or local optimal. Examples are k-means, graph theoretic, and expectation maximization. Several Collaborating filtering approaches based on partitional clustering techniques (including k-means clusterin) have been proposed [49, 54, 64].

2.2. Hard vs. Soft Clustering

Generally clustering is divided into hard and soft clustering.

- **Hard clustering (exclusive clustering)**: In hard clustering [57] each object belong to exactly one cluster, there is no uncertainty in cluster membership of an object. Object is allocated to only the cluster with which it has the greatest level of similarity [60]. K-means (Hard C means) is an important and well known hard clustering technique.

- **Soft clustering (Overlapping clustering)**: In soft clustering [57] each object belongs to two or more clusters with different degrees of membership. Instances on the boundaries between several clusters do not fully belong to one of the clusters, rather they are given membership degrees between 0 and 1 indicating their partial membership. Fuzzy C-means is very well known soft clustering technique.

2.3. Fuzzy C-means Clustering (FCM)

Fuzzy logic is different from the traditional logic methods where exact results are expected, rather fuzzy logic is used in the problems where the results can be approximate. Therefore, fuzzy logic is well suited for clustering problems, because the notion of clustering is to group similar objects together by some degree of closeness. One of the well known approach of fuzzy classification is Fuzzy C-means (FCM) clustering.

FCM produces soft partition for a given dataset by allowing the data elements to practically belong to multiple clusters. This technique was developed by Dunn [15] in 1973 and later on improved by Bezdek [5] in 1981 . The main purpose of FCM is to divide data elements that populate some multidimensional space into definite number of clusters, with the objective to find out centroids that minimizes the dissimilarity function between the clusters. The centroids (cluster centers) are calculated as the mean of all points, weighted by their likelihood of belonging to that cluster.

In FCM, a data element does not belong to exactly one cluster rather it is allowed to gradually change the membership for every cluster measured as degrees in (0,1). FCM employs fuzzy partitioning to assign each data element to multiple clusters with different membership grades. Let, the sample set be $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$ is provided, FCM divides it into c groups with centers as c_j (j= $\{1, 2, \cdots, c\}$), and the goal is to minimize the objective function, which is:

$$\mathrm{J}_c = \sum_{j=1}^{c} \sum_{i=1}^{n} u_{ij}^{a} \|x_i - c_j\|^2, 1 \leq a \leq \infty\,, \tag{1}$$

where, $u_{ij} \in [0, 1]$ represents the membership of i^{th} data element to j^{th} cluster center. The initial value of c_j (j^{th} cluster center) is selected randomly. Fuzziness is controlled by a which is any real number greater than 1. Cluster center c_j and membership function u_{ij} keep on updating until an optimize objective function is met.

$$\mathrm{u}_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|x_i - c_i\|^2}{\|x_i - c_k\|^2} \right)^{2(a-1)}} \tag{2}$$

$$\mathrm{c}_j = \frac{\sum_{i=1}^{n} (u_{ij})^a x_i}{\sum_{i=1}^{n} (u_{ij})^a} \tag{3}$$

FCM algorithm follows the given steps.

1. First step in FCM is to select number of clusters c.
2. Initially, cluster centers c_j are chosen randomly.
3. Membership function u_{ij} for each data element is computed for each cluster by equation 2.
4. Repeat the procedure until converged.
    - Compute the updated cluster center c_j by the equation 3.
    - Compute the degree of membership u_{ij} for each data element by equation 2.

Cluster center is initially selected randomly and then this procedure continues to adjust the cluster center and the degree of membership for each sample. The goal is to be converged to a saddle point of J_c or a local minimum. Performance of FCM is dependent upon initial selection of centroids, it can provide better cluster than basic k-means if the randomly chosen centers are well separated but this can not be guaranteed all the time [**?**].

FCM has become the very well-known method in cluster analysis and it performs really well in certain clustering

problems. Several researchers [34, 57, 60, 61, 62**?**] employ FCM in their research work. [61] applies FCM on Netflix dataset by adjusting objective function to directly minimize RMSE (Root mean square error) which is used to measure accuracy, in Netflix competition. But one of the limitation of FCM and its variants is that they don't perform well in high dimensional space and have considerable trouble in noisy environment [62]. A good clustering algorithm should be robust and able to tolerate these situations that often happen in real application systems.

2.4. Expectation-maximization (EM) Algorithm

The EM algorithm is an iterative method, introduced by Dempster et.al. [14] in 1977, for computing the maximum likelihood parameters of a model for incomplete data. EM algorithm iterates to recognize the expectation (E) step and the maximization step (M-step). The goal of (E) step is to computes the expectation of the log-likelihood evaluated using the current estimate for the parameters while maximization (M) step computes parameters maximizing the expected log-likelihood.These parameter-estimates are then used to determine the distribution of the latent variables in the next (E) step [14].

Given procedure is followed in EM algorithm.

1. Initialization: Randomly select value for the parameter θ that is $\theta^0 = \{\mu_1^0, \mu_2^0, \cdots, \mu_k^0\}$,

$$\theta_k^0 = \mu_k^0 \ , \tag{4}$$

   where, θ^0 is the estimate at 0^{th} iteration, μ is the mean, k is the current number of Gaussians and σ is the standard deviation.

2. Expectation (E) step: Estimate the best value for the hidden variables Z_{ij} using given parameter θ values, $\theta^t = \{\mu_1^t, \mu_2^t, \cdots, \mu_k^t\}$

$$\mathrm{E(Z}_{ik}) = \frac{exp\left[-\frac{(x_i-\mu_k^t)^2}{2\sigma^2}\right]}{\sum_{j=1}^{k} exp\left[-\frac{(x_i-\mu_k^t)^2}{2\sigma^2}\right]},$$

   where, t is the number of the iteration, $E(Z_{ik})$ is the expected value for the hidden variables, k is the dimension, σ is the standard deviation.

3. Maximization (M) step: Use the computed value of $E(z_{ik})$ to get better estimate of the parameters θ.

$$\mu_k^{t+1} = \frac{\sum_{i=1}^{n} E(Z_{ik}) x_i}{\sum_{i=1}^{n} E(Z_{ik})} \tag{5}$$

4. Convergence step: Repeat steps 2 and 3 until convergence, if $\|\theta^{t+1} - \theta^t\| < e$, stop ; otherwise, go to step 2.

A research [14] shows that each of the two steps (expectation and maximization) monotonically increase the probability of the data. But in the initialization step, the parameter values are selected randomly. Effective selection of parameter values in step 1 can yet improve the performance of EM as shown in Table 9.

2.5. k-means Clustering

Literature study shows that various methods have been proposed for solving clustering problems. k-means clustering is one of the classical and most widely used clustering algorithms developed by Mac Queen in 1967. This approach is a partitional clustering algorithm which divides the whole dataset in k disjoint clusters. It is famous for handling large datasets and its speedy convergence to local optimal. In k-means (outlined in **Algorithm 1**), firstly k initial points are chosen where k is a parameter defining the number of clusters to be sought and this parameter is defined at the start. These randomly chosen initial points are taken as cluster centers and then all the remaining dataset is scanned and each data point is allocated to closest cluster based on Euclidean distance matrix. Meanwhile the mean of all the clusters is calculated and cluster centers are updated to mean value. Subsequently this whole process is repeated with new centroid values and all the points are reassigned to new clusters. The update in centroid value is based on assignment of any new data point to the cluster or removal of any data point from the cluster. Centroid's value keeps on updating after each iteration and this process continues till there is no change in any of cluster centers i.e. cluster has converged.

2.6. Initial Centroid Selection in k-means Clustering

k-means [29] is highly unstable in initial cluster centers and this inherent limitation of k-means highly affects its efficiency. k-means randomly chooses initial centers therefore it does not guarantee to produce unique clustering results. Initial centroid selection not only influences the efficiency of the algorithm but also the number of iterations desired to run the original k-means algorithm [45]. Though k-means is known for its intelligence to cluster large datasets but its computation complexity is very expensive for massive data sets [45]. P. S. Bradley et. al. [8] put forth the concept of choosing initial centroids to resolve scalability issue. Because of the importance of initial centroid selection and k-means complexity, various methods have been proposed in the literature to enhance the accuracy and efficiency of k-means clustering with better centroid selection approaches.

Arthur and Vassilvitskii [4] proposed an algorithm called k-means++ which exploits probabilistic approach for selecting initial seeds and generates better quality clusters, as compared to classical k-means algorithm (Algorithm 1). It guarantees to produce accurate and speedy solution which is $O(logk)$ competitive to the optimal k-means solution ($O(k \times n \times itr)$, which is linear in the number of users

being clustered (n), number of clusters (k), and number of iterations (itr)). The initial centroid is chosen uniformly at random as in k-means but rest of the centroids are selected based on the probability proportional to the shortest distance from all existing centroids. Arthur et. al. report that k-means++ efficiently produces better quality both for synthetic and natural datasets. Shindler [56] reviewed many clustering algorithm and reported k-means++ as the most successful method for defining initial seeds of k-means clustering. The problem [56] with k-means++ algorithm is that it is inherently sequential.

Abdul Nazeer et al. [44] proposed an algorithm to solve time complexity and initial centroid selection issue of original k-means. The author modified two phases of k-means, where in phase one, pair-wise distance of data points are calculated and closest data point forms the cluster. Threshold value for number of data points in a cluster is defined. Mean value of resulting clusters are taken as initial centroids. The second phase is variant of [30] which takes the resulting initial centroids of phase one as input. The algorithm proposed by [44] produces better quality clusters in less amount of time. This algorithm has further been enhanced in [45] by using heuristic approach. Firstly, all the data points are sorted in ascending order and then divided in k sets. The mean values of these sets are calculated and taken as initial centroids. Heuristic for multidimensional data is also proposed in [45]. Complexity of this phase is reduced to $O(nlogn)$.

A method for finding initial centroids is proposed by Fang Yuan et al. [16]. In this method, centroid are produced systematically, which are consistent with distribution of data. Though no improvement in time complexity is proposed but it produces better quality clusters. Fang YuanFang Yuan et al. [65] provide comparison of 14 k-means cluster initialization methods, which come under two main categories—synthetic starting points and actual sample starting points.

Synthetic initial point refers to a point, which is not associated with any of the actual point in the dataset. The main algorithms for synthetic initial points, discussed in [65] generally divide the dataset into different partitions (where number of partitions has no significant influence on clustering results). Then for each partition, values for each feature acts as initial point of that partition and ultimately different starting point of the features are combined to generate initial seeds for k-means clustering. They report that scrambled midpoints is best synthetic method that follows the same steps mentioned above, with the addition that it takes mid points of each partition for each feature as initial points and then randomly selects any partitions midpoint as initial starting point for final clustering.

Actual sample starting point, another category discussed in [65], selects an initial point that actually resides in the dataset. Random, feature value sums and breakup, are methods discussed in [65] and it is reported that breakup method outperform rest of two methods, as it seeks to break up the most populous cluster into two smaller cluster for choosing initial seeds for k-means clustering. They also compared the best methods of synthetic and actual categories and concluded that scrambled midpoint performs much better than breakup.

J.M. Pena et al. [50] compared four k-means cluster initialization approaches—Random, Forgy, MacQueen and Kaufman. They performed extensive experiments on three real world dataset (Iris, Glass, Ruspini) to compare aforementioned initialization approaches in term of quality of cluster they produce and sensitivity of k-means towards initial starting conditions of these methods. They reported that Kaufman initialization algorithm [47] performs the best among the discussed methods. It chooses k seeds while the first seed is the most centrally located data point in the dataset. The rest of points are chosen based on heuristic rule of having highest number of neighbours in the multidimensional space.

Arai et.al. [3] uses both k-means and hierarchical clustering for centroid initialization. It exploit clustering results of k-means algorithm and transform them by combining with hierarchical algorithm, to produce better quality clusters by finding better initial cluster centers for k-means. It performs well for complex clustering problems with large datasets having multidimensional attributes. It is a computationally expensive approach, as it performs k-means clustering followed by hierarchical clustering. Divisive Correlation Clustering Algorithm (DCCA) is proposed in [6] for grouping of genes. DCCA does not take value of k and initial centroids as input for clustering. The time complexity and repairing cost from any misplacement of [6] are too high.

K-Modes method is another variation of the k-means clustering algorithm [28], Kmode2001. In K-Mode method mode is taken as cluster centroid rather than mean. Huang et al. [28] used both k-means and K-Modes to cluster the data and proposed K-prototype algorithm for this purpose.

Cluster Center Initialization Algorithm (CCIA) is proposed by Shehroz and Ahmad [31] to solve the initialization problem. It starts with taking the mean and standard deviation of object attributes, and then divides the dataset into definite partitions. k-means and density-based multi scale data condensation are used to find out the similarity in the partitions, initial clusters are selected afterwards based on similarity results.

Ivan Cabria [11] proposed Mean Shift initialization method, which does not depends upon number of clusters (i.e. k). It finds the modes of the underlying probability density function of the given data, and these modes are a better choice as initial cluster centers for k-means. Authors performed experimental study on proposed algorithm and other classical algorithm using two real-life problems—Facility Location and Molecular Dynamics—with very large amounts of data. Experiments report that it outperforms other clustering algorithms in term of clustering performance.

Maitra et al. [37] present a detailed evaluation of eleven widely used and well performing k-means initialization

methods on nine standard datasets. They report that their research work on real datasets do not provide definite results so they evaluated experimental performance of initialization methods by conducting systematic and comprehensive simulation study. Maitra et al. [37] state that [8, 40, 41] outperforms all the rest methods in minimizing objective function—WSS (Within group Sum-of-Squares) as well as in achieving true grouping.

G. Tzortzis et al. [59] proposed the MinMax k-means algorithm with the objective to minimize the maximum intra-cluster variance. They claim that MinMax algorithm provides high quality clusters irrespective of initial selection, by limiting the occurrence of large variance clusters. Experiments verify the effectiveness and robustness of proposed approach over traditional k-means.

Recently C. Zhang et al. [68] proposed an enhanced method that improves centroid selection and determines value of k for k-means clustering. They performed simulated analysis on UCI datasets and reported that [68]'s method improves the accuracy and efficiency of clustering process because of its stability and capability to avoid the affect of noisy data. Specifically, this method finds the distance between all data points and computes density for each point. A density threshold is defined at the start, and all the points that realized density threshold are placed in a centroid set. The first seed is selected as the most populous point in the centroid set and all the rest are selected by decreasing density with the rule that they are at largest distance from already selected centroids. Data points are assigned to closest centroid and centers are updated based on an improved and adjusted heuristic function, by assigning different weights to data points as per their distance from centroid.

There is very limited work for choosing seeds in k-means clustering for recommender system domain. As the conventional k-means clustering algorithm [29] randomly selects the k initial centroids, an important research question would be, *"how does the quality of clusters and recommendations vary with the choice of different initial centroids in Clustering-Based CF recommender systems"*? We investigate how to improve the quality of clusters and recommendations focusing on the aforementioned key issue.

3. Proposed Centroid Selection Approaches

In this paper, we have implemented a series (eighteen) of novel centroid selection approaches in k-means clustering for improving the recommendation process for recommender systems. We have applied these selection approaches along with traditional k-means for comparing their performance. The algorithms present the centroid selection procedure for k-means clustering. After selecting k seeds, next steps are followed as per Algorithm 1, to accomplish k-means clustering.

Algorithm 1, denoted by $KMeans$, outlines the centroid selection procedure of traditional k-means algorithm [29] which selects k users from training set uniformly at random. We have implemented k-means to benchmark our work with the conventional approach and present the improvements in cluster quality and recommendation quality.

Algorithm 2, denoted by $KMeansPlus$, selects the first centroid totally at random (as in conventional k-means). After that $KMeansPlus$ keep choosing seeds having maximum distance from all existing centroids, until k centroid are chosen.

In **Algorithm 3**, denoted by $KMeansPlusPlus$, represents a k-means variant, which selects k centroids from training set by using k-means++ algorithm [4]. The first centroid is chosen uniformly at random as in k-means but rests of all centroids are selected based on the probability proportional to maximum distance from all existing centroids.

Algorithm 4, denoted by $KMeansDensity$, chooses initial centroids based on nearest neighbour density. All the initial centroid are well-separated from each other as well as they possess large number of neighbours in multidimensional sphere [37]. In step 2, we find out average pair wise Euclidean distance d_1 between all the users. In step 3, we use d_1 as threshold to find out neighbouring points, for all the users. Then in step 4, we sort the users' list based on the number of neighbours. In step 5, we choose first centroid centered at x_1, which has maximum number of neighbours in a multidimensional sphere with the radius d_1. All the rest centroids are chosen based on decreasing number of neighbours with the rule that they are at d_1 distance from the centroid.

Algorithm 5, denoted by $KMeansVariance$, proposes an initialization approach that selects the centroids that are at varying distances from the overall mean. In step 2, we find out the mean rating provided by all users in the dataset and in step 3, we sort the dataset based on Euclidean distance from the mean value — average user ratings. In step 5, for cluster $L : L = \{1, 2, \cdots k\}$, the $1 + (L-1) * M/k$ point is selected to be its initial cluster centre, where K sample points are chosen as the initial cluster centers from M datapoints [37]. Using this initialization process, it is guaranteed that no cluster will be empty after the initial assignment.

Algorithm 6, denoted by $KMeansVariance^{AvgPairWise}$, presents a variant of Algorithm 5, which follows all the steps same as Algorithm 5 except that in step 3, it uses mean value as average pair-wise distance between all the users.

In **Algorithm 7**, denoted by $KMeansVariance^{Version}$, propose another variant of Algorithm 5 that only differs in step 4, where we measure standard deviation — for finding initial centroids — from overall mean rating provided by all users.

Algorithm 8, denoted by $KMeansQuantiles$, represents a variant of k-means which chooses initial centroids as quantiles of the given datasets [37]. These quantiles correspond to equal increments in probabilities.

Algorithm 9, denoted by $KMeansPlus^{Density}$, is a modification of Algorithm 4. It initializes the highest den-

sity point as the first centroid and automatically computes minimum distance that separates the centroids based on highest density point, which is close to maximum number of other points in the data set [33]. In step 2, we find out average pair wise Euclidean distance d_1 between all the users. In step 3, we find out sum of distances of all the users from x_i denoted by Sum(i). Then in step 4, we sort the users' list based on value of Sum(i) and find out highest density point (i.e. user with minimum value of Sum(i)). In Step 5, we choose first centroid centered at x_1, which has maximum number of neighbours in a multidimensional sphere with the radius d_1. All the rest centroids are chosen based on decreasing number of neighbours with the rule that they are at d_1 distance from the centroid.

Algorithm 10, denoted by $KMeansSortedDistance$, uses a simple approach of sorting all user ids based on the distance from average rating provided in the dataset. We calculate the distances in step 2 and sort them in step 3. First k points are chosen as initial centroids in step 4.

Algorithm 1 : $KMeans$, Selects k users as centroids from the dataset
Input: $\mathcal{U}$, training users; k, the number of clusters
Output: $\{ c_1, c_2, \cdots c_k \}$, k centroids

1: Define desired numbers of clusters, k.
2: Choose the k users uniformly at random from $\mathcal{U}$, as initial starting points.
3: Assign each user to the cluster with nearest centroid.
4: Calculate mean of all clusters and update centroid value to the mean value of that cluster.
5: Repeat step 3 and 4, till no user changes its cluster membership or any other convergence criteria is met.
6: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 2 : $KMeansPlus$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k.
2: Select initial centroid c_1 from $\mathcal{U}$, uniformly at random.
3: **repeat**
4: Choose next centroid c_i where $c_i = u' \in \mathcal{U}$ with probability:

$$\text{Prob} = \frac{\text{dist}(u')^2}{\sum_{u \in \mathcal{U}} \text{dist}(u)^2}.$$

5: **until** k centroids are found
6: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

3.1. Exploiting Power Users Concept

The users who have rated a large number of items in a recommender system [27] are referred to as power users. In some of our approaches we have exploited power users — for clustering user-item rating matrix. Several researchers have utilized this concept to generate scalable CF-based recommendations, for example [66] focused on issues in tag-based recommender systems. They proposed that by using power users for recommendation, accurate and scalable recommendations can be generated. They claimed that in social tagging websites like CiteULike, the users who give large number of tags are referred to as *leaders* [66] and those who follow these tags are called *followers*. The opinions of leaders, if taken into account, assist to generate better recommendation—speedy convergence and better clustering. In this work, u_p denotes power users—users with maximum number of ratings in the training set. We normalize the number of rating assigned by user u, by dividing it by the rating assigned by power user u_p, instead of using raw rating count. Formally:

$$\mathbb{P}(u) = \frac{|\mathcal{I}_u|}{|\mathcal{I}_{u_p}|}, \tag{6}$$

where $|\mathcal{I}_u|$ and $|\mathcal{I}_{u_p}|$ refers to the number of items rated by user u and u_p respectively. Suppose $\mathcal{U}^{power} = \{ u_1, u_2, \cdots, u_z \}$ is the set of z power users with highest value for $\mathbb{P}(u)$ (that is $\mathbb{P}(u_m) > \mathbb{P}(u_n) : \forall_{u_m \in \mathcal{U}^{power} AND\, u_n \notin \mathcal{U}^{power}}$); $C = \{c_1, c_2, \cdots c_k\}$ represents k centroids, while $G = \{g_1, g_2, \cdots g_k\}$ is the set of k clusters. The set of user-item pairs rated in any cluster g_j with centroid c_j (that is $(r_{i,u}|(i,u) \in \mathcal{D}_{c_j})$), is represented by $\mathcal{D}_{c_j}$; the rating assigned to an item i by a centroid c_j is denoted by r_{i,c_j}, that is $r_{i,c_j} = \frac{1}{|\mathcal{D}_{c_j}|} \sum_{i,u \in \mathcal{D}_{c_j}} r_{i,u}$ and dist(u) represents the shortest distance from any user u to the closest centroid which is already chosen.

In **Algorithm 11**, denoted by $KMeansPlus^{Power}$, presents a k-means variant, which utilizes power users and applies k-means++ algorithm on power users only. First of all power users ($\mathcal{U}^{power}$ users with $\mathbb{P}(u) > \text{pow}_{\text{thr}}$)[1] are identified from steps 2 to 7. After that k-means++

Algorithm 3 : $KMeansPlusPlus$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k.
2: Select initial centroid c_1 from $\mathcal{U}$, uniformly at random.
3: **repeat**
4: Choose next centroid c_i where $c_i = u' \in \mathcal{U}$ with probability:

$$\text{Prob} = \frac{\text{dist}(u')^2}{\sum_{u \in \mathcal{U}} \text{dist}(u)^2} \propto \text{dist}(u')^2.$$

5: **until** k centroids are found
6: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

[1] The value of pow_{thr} can be found using training set.

algorithm is applied which exploits these power users as candidate centroids while choosing k centroids.

Algorithm 12, denoted by $KMeansPlus^{ProbPower}$, represents a k-means variant ,which seeks to find k centroids which have probability proportional to distance and the number of ratings as shown in step 4 of Algorithm 12. The motivation behind this variant is to get potential benefits (increased coverage and reduced error) by choosing those users as centroid, who have maximum distance, (i.e. minimum similarity) with the existing ones and also who have provided maximum number of ratings.

Algorithm 13, denoted by $KMeansPlus^{LogPower}$, represents a k-means variant, which utilizes power users to find k centroids which have probability proportional to distance and the log of similarity with existing top power user, as shown in step 4 of Algorithm 13. Where $p(x)$ can be computed by dividing, the number of movies seen by active user by the number of movies seen by power user.

Algorithm 4 : $KMeansDensity$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k.
2: Find the average pair wise Euclidean distance, denoted by d_1, as follows:

$$d_1 = \frac{1}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \|x_i - x_j\| .$$

3: Find out the number of neighbours at distance d for all users, where $d \leq d_1$
4: Sort all the users based on the number of neighbors in multidimensional sphere.
5: Select initial centroid c_1 from U, as the point x_i with the largest number of points within a multidimensional sphere with radius d_1 that is centered at x_i.
6: The remaining seeds are chosen by decreasing density, with the restriction that all remaining seeds must be d_1 distance away from all previous seeds
7: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

3.2. Exploiting Data Distribution

Whenever we collect data for experimentation, typically there lies some degree of variability in that data. So the raw data does not generate better results in certain cases. For that fitting any appropriate statistical distribution to the data is a good idea. We apply some of the data distributions to our dataset and compare their results.

In **Algorithm 14** and **15** we applied normal distribution over the dataset and arbitrarily select k centroids from the distributed dataset. In **Algorithm 14**, denoted by $KMeansNormal^{Users}$, we exploit average rating provided by all the users in the dataset as the mean value of normal

Algorithm 5 : $KMeansVariance$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k.
2: Take mean rating of all the dataset (uids).
3: Sort all uids based on the Euclidean distance from the mean value.
4: **repeat**
5: Let $L : L = \{1, 2, \cdots k\}$. Choose seed using the following formula:

$$s_L = x_{1+(L-1)*\frac{M}{k}}$$

6: **until** k centroids are found
7: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 6 : $KMeansVariance^{AvgPairwise}$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k.
2: Take mean rating of all the dataset (uids).
3: Sort all uids based on the Euclidean distance from the mean—average pair-wise distance between all the users.
4: **repeat**
5: Let $L : L = \{1, 2, \cdots k\}$. Choose seed using the following formula:

$$s_L = x_{1+(L-1)*\frac{M}{k}}$$

6: **until** k centroids are found
7: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 7 : $KMeansVariance^{Version}$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k.
2: Take mean rating of all the dataset (uids).
3: Sort all uids based on the Euclidean distance from the mean value.
4: Use Standard Deviation to choose k seeds lying at varying distance from overall average users' rating in the dataset.
5: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 8 : *KMeansQuantiles*, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k.
2: **repeat**
3: Let $L : L = \{1, 2, \cdots k\}$. Choose seed using the following formula:

$$\mathrm{s_L} = \mathrm{x}_{1+(\mathrm{L}-1)*\frac{\mathrm{M}}{\mathrm{k}}}$$

4: **until** k centroids are found
5: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 9 : $KMeansPlus^{Density}$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k.
2: Find the average pair wise Euclidean distance, denoted by d_1, as follows:

$$\mathrm{d}_1 = \frac{1}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \|x_i - x_j\| .$$

3: Compute the sum of distances from x_i^{th} position to all other points, as Sum(i).
4: Sort Sum(i), and find highest density point x_h, which is the minimum value of Sum at index h.
5: Select initial centroid c_1 to be x_h
6: **repeat**
7: Set $\mathrm{d}(x_i)$ to be the distance between x_i and the nearest point in C, for each point x_i.
8: Find the sum of distances of first m/k nearest points from the x_h, denoted by y.
9: Find the unique integer i so that

$$d(\mathrm{x}_1)^2 + d(\mathrm{x}_2)^2 + \cdots + d(\mathrm{x}_i)^2 \geq \mathrm{y} > d(\mathrm{x}_1)^2 + d(\mathrm{x}_2)^2 + \cdots + d(\mathrm{x}_{i-1})^2$$

10: Choose x_i as next centroid
11: **until** k centroids are found
12: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 10 : *KMeansSortedDistance*, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k.
2: Find the distance of all uids (user ids) from the global average rating- ratings provided by all the users to all the movies.
3: Sort these distances in ascending order.
4: Choose first k data points from sorted list as initial centroids.
5: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 11 : $KMeansPlus^{Power}$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k
2: $\mathcal{U}^{power} = \emptyset$
3: **for all** $u \in \mathcal{U}$ **do**
4: **if** $\mathbb{P}(u) > \mathrm{pow_{thr}}$ **then**
5: $\mathcal{U}^{power} = \mathcal{U}^{power} \cup u$
6: **end if**
7: **end for**
8: $\{ c_1, c_2, \cdots c_k \} = KMeansPlusPlus\ (\mathcal{U}^{power}, k)$
9: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 12 : $KMeansPlus^{ProbPower}$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k
2: Select the initial centroid c_1 to be u_p.
3: **repeat**
4: Select the next centroid c_i where $c_i = u' \in \mathcal{U}$ with the probability:

$$\mathrm{Prob} = \frac{\left(\frac{\mathrm{dist}(u')^2}{\sum_{u \in \mathcal{U}} \mathrm{dist}(u)^2} + \frac{\mathbb{P}(u')^2}{\sum_{u \in \mathcal{U}} \mathbb{P}(u)^2} \right)}{2} .$$

5: **until** k centroids are found
6: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 13 : $KMeansPlus^{LogPower}$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k
2: Select the initial centroid c_1 to be u_p.
3: **repeat**
4: Select the next centroid c_i where $c_i = u' \in \mathcal{U}$ with the probability:

$$\text{Prob} = \text{dist}(u) + \log[\frac{1}{p(x)} + 1]$$

5: **until** k centroids are found
6: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 14 : $KMeansNormal^{Users}$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k
2: Construct a Gaussian distribution of the dataset by providing average users rating as mean of the distribution.
3: Select k centroid randomly from distributed dataset.
4: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 15 : $KMeansNormal^{Movies}$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k
2: Construct a Gaussian distribution of the dataset by providing average movies rating as mean of the distribution.
3: Select k centroid randomly from distributed dataset.
4: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 16 : $KMeansPoisson$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k
2: Apply Poisson distribution over the dataset.
3: Select k centroid randomly from distributed dataset.
4: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 17 : $KMeansHyperGeometric$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k
2: Apply Hypergeometric distribution over the dataset.
3: Select k centroid randomly from distributed dataset.
4: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 18 :$KMeansUniform$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k
2: Apply Uniform distribution of the dataset with the given minimum and maximum range as 0 and k respectively.
3: Select k centroid randomly from distributed dataset.
4: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 19 : $KMeansUniform^{Version}$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k
2: Apply Uniform distribution of the dataset with the given range as minimum and maximum distance of users from the centroid.
3: Select k centroid randomly from distributed dataset.
4: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

Algorithm 20 : $KMeansLog$, Choose k users from the dataset, as centroids
Input: $\mathcal{U}$, users in training set; k, total number of clusters;
Output: k centroids, $\{ c_1, c_2, \cdots c_k \}$

1: Define desired number of clusters, k
2: Apply Logarithmic distribution over the dataset.
3: Select k centroid randomly from distributed dataset.
4: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids

distribution curve, while in **Algorithm 15**, denoted by $KMeansNormal^{Movies}$, we utilized the average rating of all the movies in the dataset as the mean value of normal distribution curve.

In **Algorithm 16**, denoted by $KMeansPoisson$, we construct Poisson distribution of the dataset, and arbitrarily select k centroids from the distributed dataset.

In **Algorithm 17**, denoted by $KMeansHyperGeometric$, we applied hyper geometric distribution over the dataset, and arbitrarily select k centroids from the distributed dataset.

In **Algorithm 18** and **19**, denoted by $KMeansUniform$ and $KMeansUniform^{Version}$ respectively, we apply uniform distribution over the dataset, and arbitrarily select k centroids from the distributed dataset.

In **Algorithm 20**, denoted by $KMeansLog$, we construct logarithmic distribution of the dataset, and arbitrarily select k centroids from the distributed dataset.

3.3. Distance measure

In Algorithm 2, 3, 12, and 13 dist function is used to measure the distance between a user and a centroid. We are using similarity instead of distance, which can be computed by measuring the similarity between a user and a centroid. As similarity and distance are inversely proportional to each other, thus the distance between two points is maximum if the similarity is zero and vice versa. Distance function can be modeled as follows:

$$\text{dist} = \begin{cases} \frac{1}{\text{sim}} & if\ \text{sim} \neq 0\,, \\ \text{MAX}_{\text{DIST}} & otherwise, \end{cases} \tag{7}$$

where MAX_{DIST} (chosen as 1000 in our case[2]) denotes the maximum distance between two points. As in Pearson correlation, the similarity result can be negative as well, but the negative similarity cannot be modeled using Equation 7. We handled this issue in centroid selection approaches, by adding 1 to all the similarity results returned by Pearson correlation (i.e. $sim(u) \leftarrow sim(u) + 1\,, \forall_{u \in \mathcal{U}}$), after that Equation 7 is applied on positive similarity results. As CF based approaches only considers the intersection of the items, which are voted by both the centroid and the active user. Thus it does not perform well when there are less common items between the centroid and the active user. We can overcome this problem by assuming some defaults votes for the items that have not been voted by a user or a centroid, and extending the correlation over the union of items rather than intersection; in order to increase the accuracy and coverage of the recommender system. The resultant distance equation is termed as PCCDV. For further information, refer to our previous work [24].

[2]Any value greater than 1 can be chosen as MAX_{DIST}.

3.4. Clustering of Dataset and Generating Recommendation

Algorithm 21 shows the clustering and recommendation approach used in this work. The user-item rating matrix is clustered into k clusters and then recommendation is provided based on the similar centroids ratings. For choosing k initial centroids, a centroid selection algorithm is desired. In step 2, CentroidSelect procedure choses any proposed centroid selection algorithm for finding k initial centroids. A loop counter is initialized in step 3, for counting number of iteration for the execution of the algorithm. In Step 5, we find out the similarity between the user and centroids; and each user is assigned to the cluster with the most similar centroid. Then we update the centroids to the set of user-item pairs that have been rated by all the users residing in that cluster, in step 6. In step 7, the loop counter is incremented and step 5 to 7 keep executing till the loop counter value is equal to itr or the clusters converges (i.e. no user changes its cluster membership). We return the clustered data, in step 9.

The centroids of the clusters contain the average ratings assigned to all the items, which are rated by the users in that cluster. Each centroid can be assumed to be the vector of length N (i.e. total number of items in the system) and can be denoted by: $c_j = \{\, r_{i_1,c_j}, r_{i_2,c_j}, \cdots, r_{i_N,c_j} \}$, where r_{i,c_j} represents the ratings assigned to an item i by the centroid c_j, mathematically: $r_{i,c_j} = \frac{1}{|\mathcal{D}_{c_j}|} \sum_{i,u \in \mathcal{D}_{c_j}} r_{i,u}$. Equation 8 is used to estimate the rating value of the item in a cluster, if no rating has been provided by users in corresponding cluster to that item.

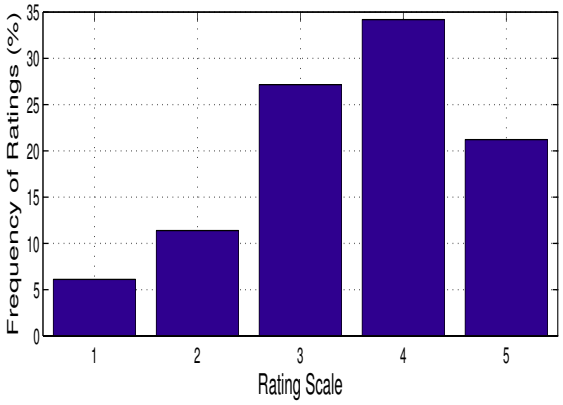
$$\text{r} = \begin{cases} \frac{1}{|\mathcal{D}_u|} \sum_{i \in \mathcal{D}_{c_j}} r_{i,u} & if\ \text{r}_{\text{i,u}} \neq \emptyset\,, \\ \frac{1}{|\mathcal{D}_{c_j}|} \sum_{i,u \in \mathcal{D}_{c_j}} r_{i,u} & if\ \text{r}_{\text{i,c}_\text{j}} \neq \emptyset\,, \end{cases} \tag{8}$$

Now all the clusters centroid contains average rating for the items in the system. In step 12, we find out the similarity between the active user and all the k centroids by Pearson correlation. Then in step 13 we identify most similar centroids, refer to as neighbours of active user. After that prediction is made for the target item and Top-N items [24] can be presented to user as recommendations.

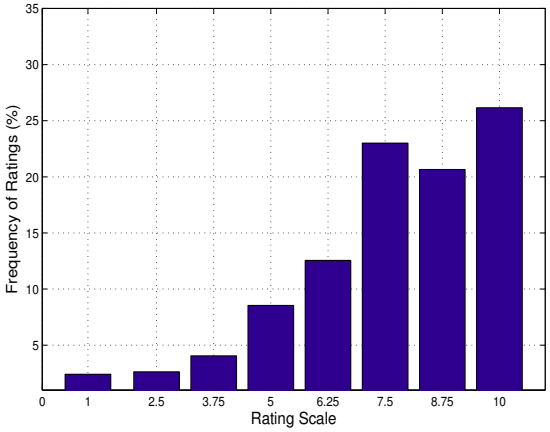
4. Experimental Setup

4.1. Datasets

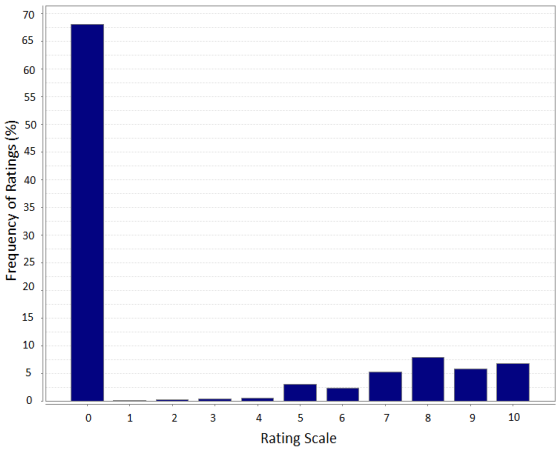
For these experiments, our dataset is made up of data captured from film, book and music recommendation websites, which are commonly used in the field of recommender systems. As they are some of largest datasets and are used frequently in literature to test recommendation algorithms, they facilitate us to measure the scalability of our algorithm as well as to benchmark our algorithm with some of the state of art algorithms. In this paper, we used MovieLens (100K and 1M ratings), FilmTrust, BookCrossing and LastFM dataset.



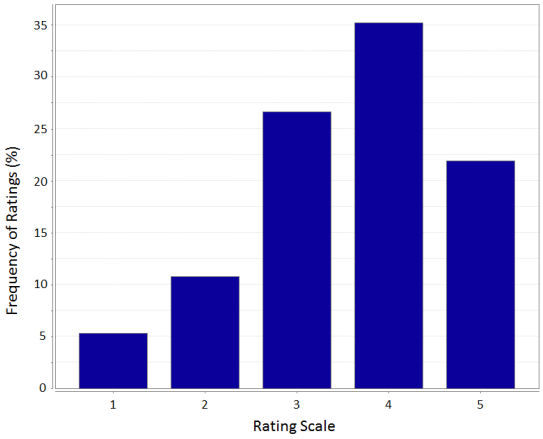
(a) Rating distribution of the SML dataset.



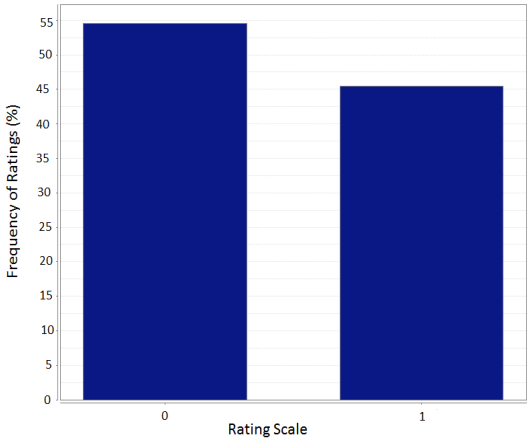
(b) Rating distribution of the FilmTrust dataset.



(c) Rating distribution of the Book-Crossing dataset.



(d) Rating distribution of the ML dataset.



(e) Rating distribution of the FM dataset.

Figure 1: (From left to right and top to bottom) Rating distribution of the MovieLens (SML), FilmTrust (FT), Book-Crossing (BC), MovieLens (ML), and LastFM (FM) datasets.

Algorithm 21 : **ClustAndRecommend**, Clusters the user-item rating matrix into k clusters and generate recommendations.
Input: $\mathcal{U}$, training users; k, the number of clusters; itr, the number of iteration for the k-means clustering algorithm
Output: G, C; the clusters with corresponding centroids

procedure CLUSTER($\mathcal{U}, k,$ itr)
  $C =$ **CentroidSelect**
  $t = 0$
  repeat
    Set the cluster g_j, for each $j \in 1, ..., k$, to be the set of users in $\mathcal{U}$ that are closer to c_j than they are to c_l for all $l \neq j$.
    Set c_j, for each $j \in 1, ..., k$, to be the centre of mass of all users in g_j, i.e.

$$c_j = \frac{1}{|g_j|} \sum_{u \in g_j} u \,.$$

    $t = t + 1$
  until (C no longer changes) OR ($t =$ itr)
  return (G, C)
end procedure

procedure *Recommend*($\mathcal{D}^{test}$)
  Use Pearson correlation to find the similarity between an active user and k other centroids using the
  Find the neighbours of the active user, i.e. l ($l \leq k$) most similar centroids, called
  Make prediction on target item using the weighted average of the ratings provided by neighbours
end procedure

- **MovieLens 100K Ratings (SML)**: GroupLens [32] currently provide three movie-rating datasets. We have utilized one of those datasets (denoted by SML in this thesis) with 100 000 ratings provided by 943 users for 1682 movies. Movies are rated on an integer scale of 1 (bad) to 5 (excellent). This matrix has sparsity of 93.7% - calculated as $\left(1 - \frac{non\ zero\ entries}{all\ possible\ entries}\right)$ which means that 6.3% of total user-item pairs have been rated. Figure1(a) shows the rating distribution of the SML dataset.

- **MovieLens 1M Ratings (ML)**: We have used another dataset (denoted by ML in this work) provided by GroupLens [32], which contains 6 040 users, 3 900 movies, and 1 000 000 ratings. Movies are rated on an integer scale of 1 (bad) to 5 (excellent). We observe that in the MovieLens dataset, the rating distribution is skewed towards rating of 4. Figure1(d) shows the rating distribution of the ML dataset.

- **FilmTrust (FT)**: This is not a pre-packaged dataset, we crawled FilmTrust website and created this dataset. The dataset gathered (on 10th March 2009) includes 28 645 ratings provided by 1 214 users for 1 922 movies. Movies are rated on a floating point scale of 1 (bad) to 10 (excellent) (with scale of 0.25). Its sparsity is 98.8%. Figure 1(b) shows the rating distribution of the FilmTrust dataset. This dataset is very sparse and imbalanced—one user might have rated one item and other might have rated dozens of items (and same is true for items as well)—and it well represents the cold-start scenarios (where we have number of new users and items in the system) [25].

- **Book-Crossing (BC)**: Book-Crossing dataset was crawled by Cai-Nicolas Ziegler [69] in August/September 2004 from the Book-Crossing community. It contains 1 149 780 ratings (explicit / implicit) provided by 278 858 users about 271 379 books referring to distinct ISBNs, while invalid ISBNs were discarded from the dataset. Ratings are provided on a scale from 1 (bad) to 10 (excellent). We performed condensation of dataset to overcome the extreme sparsity problem and to generate meaningful results. For that we filtered the users and books with less than 25 overall mentions. Resulting dataset contains 200 093 ratings provided by 5 892 users about 5 610 books. Figure 1(c) shows the rating distribution of the Book-Crossing dataset.

- **LastFM (FM)**: This dataset contains music artist listening information taken from Last.fm online music system [12]. This dataset contains 92 834 user-listened artist relations for 1 892 users and 1 7632 artists. User-listened artist relations provides a listening count for each [user, artist] pair. We have utilized listening count to find out binary scale rating (*like/dislike*),

by providing a threshold value (in our case 300), for each [user, artist] pair. Hence, if a listing count for a particular [user, artist] pair is greater than 300, we assign it *like* class and *dislike* class otherwise. Figure 1(e) shows the rating distribution of the resultant dataset.

4.2. Metrics

Mean Absolute Error (MAE) has been extensively used in many research projects, such as [9, 19, 22, 23, 51, 52, 53, 64]. It computes the average absolute deviation between predicted rating provided by a recommender system and a true rating assigned by the user. It is computed as:

$$\text{MAE} = \frac{1}{|\mathcal{D}^{test}|} \sum_{i=1}^{|\mathcal{D}^{test}|} |p_i - a_i|,$$

Where $|\mathcal{D}^{test}|$ is the total numbers of ratings provided by the test set, p_i is predicted rating provided by recommender system and a_i is the actual rating assigned by user to item i. Recommender systems intend to reduce MAE score by minimizing the difference between predicted and actual rating of an item.

Moreover, we used *coverage*, which describes the number of user-item pairs that a recommendation algorithm can make prediction for. Some authors denote this metric by prediction coverage [27]. The coverage metrics has been used in [20, 22, 25]. There are a number of metrics that are more specifically designed to measure how well a recommender classifies good quality (relevant) items. These include the *ROC-sensitivity* and $F1$ *measure* and have been used in projects, such as [22, 25, 52, 53]. The details of using coverage, ROC, and F1 metrics for recommender system settings can be found in our previous work [25].

4.3. Evaluation methodology

In this work, we have randomly divided the dataset into training and testing set by performing 5-fold cross validation and reported average results. Specifically, for each user, 80% randomly divided movies (rated by them) are chosen as training set and the rest (20%) as the test set. The training set is further divided into training (80%) and validation set (20%) to train the parameters.

5. Results and Discussion

In this section, we present the performance comparison of aforementioned approaches for different centroid selection approaches in terms of MAE in recommendation, cluster quality, cluster convergence, cluster building time, and recommendation coverage. We benchmark the results with conventional k-means algorithm (i.e. Algorithm 1).

5.1. Performance comparison in terms of recommendation MAE

To compare the performance of different centroid selection approaches, in term of recommendation we measure MAE, ROC-Sensitivity, Precision, Recall, and F1 for all the aforementioned algorithms. For instance, we present the results computed under optimal parameters for MAE, in Table 2.

The results—over SML dataset in Figure 2(a)—show that Algorithm 12 (i.e. $KmeansPlus^{ProbPower}$) and Algorithm 13 (i.e. $KMeansPlus^{LogPower}$) perform the best. Figure 2(b), detailing the results over FT dataset, depicts that the MAE of Algorithm 13 (i.e. $KMeansPlus^{LogPower}$) outperforms the rest. Moreover, the accuracy of Algorithm 12 (i.e. $KmeansPlus^{ProbPower}$) is comparable to Algorithm 13 (i.e. $KMeansPlus^{LogPower}$).

Results over BC dataset in Figure 2(c) show that most of centroids selection algorithm generate results better than k-means but Algorithm 16 (i.e. $KMeansPoisson$) performs the best. We can also find out that Algorithm 12 (i.e. $KmeansPlus^{ProbPower}$) performs the best over ML dataset as shown in Figure 2(d). Moreover, the results of Algorithm 4 (i.e. $KMeansDensity$) are comparable to the best approaches for this dataset. Figure 2(e) presents that Algorithm 20 (i.e. $KMeansLog$) gives outstanding result for FM dataset. The results of Algorithm 4 (i.e. $KMeansDensity$) are comparable to the best approach for this dataset.

5.2. Performance comparison in terms of cluster quality

We proposed various centroid selection approaches in this work, with the statement that cluster quality (that is total within-cluster similarity between each cluster centroid and users) is dependent upon initial centroid selection. To prove our statement, we present results of different centroid selection approaches at varying number of iterations. For each user $u \in \mathcal{U}$ in the training set, the total within-cluster similarity is measured as follows:

$$\text{TotalSim}(\mathcal{U}, G) = \sum_{g_j \in G} \sum_{u \in \mathcal{U}^{g_j}} \text{sim}(u, c_j), \tag{9}$$

Where c_j denotes the centroid of the cluster g_j, while $\mathcal{U}^{g_j}$ are the number of users in the corresponding cluster g_j and G denotes the total number of clusters. Table 3 depict the total within-cluster similarity between each user and the cluster it belongs to, for SML dataset. We observe that within-cluster similarity of Algorithm 10 ($KMeansSortedDistance$) is better than all of rest for the SML dataset. While for FT dataset within-cluster similarity of Algorithm 3 ($KMeansPlusPlus$) is highest while that of Algorithm 4 ($KmeansDensity$) is nearly the same, as shown in Table 4. Results of BC and ML-1M datasets in Table 5 and 6 shows that Algorithm 13 ($KMeansPlus^{LogPower}$) results in highest within-cluster similarity for both datasets. While

Table 1: Tuning the distance measure in term of MAE and coverage to select the optimal one for subsequent experiments. The number of clusters, iterations and neighbours are set to the optimal value. The best results are shown in bold font. PCCWithDefault represents the Pearson Correlation distance measure where default votes are used to overcome sparsity, PCCWithoutDefault represents the Pearson Correlation distance measure without default votes, VSWithDefault represents the Vector Similarity distance measure where default votes are used to overcome sparsity, and VSWithoutDefault represents the Vector Similarity distance measure without default votes.

	MAE					**Coverage**				
Distance Measure	**SML**	**FT**	**BC**	**ML**	**FM**	**SML**	**FT**	**BC**	**ML**	**FM**
PCCwithDefault	**0.745**	**1.493**	2.912	0.867	0.390	99.785	**95.951**	**97.325**	**100.0**	**100.0**
PCCwithoutDefault	0.754	1.495	**2.835**	**0.863**	**0.376**	99.765	95.790	97.497	**100.0**	98.179
VSWithDefault	0.773	1.508	2.845	0.882	0.381	**99.795**	95.812	98.074	**100.0**	97.463
VSWithoutDefault	0.769	1.498	2.844	0.897	0.382	99.775	95.812	98.062	99.999	97.075

Table 2: Inspecting the performance of different centroid selection approaches in terms of MAE and coverage, over the test set. The number of clusters, iterations and neighbours are set to the optimal one. Also the optimal distance measure is used to find the similarity between a user and the centroid. The best results are shown in bold font.

	MAE					**Coverage**				
Centroid Selection	**SML**	**FT**	**BC**	**ML**	**FM**	**SML**	**FT**	**BC**	**ML**	**FM**
$KMeans$	0.745	1.493	2.885	0.863	0.376	99.785	96.04	97.497	**100**	98.179
$KMeansPlus$	0.745	1.484	2.882	0.873	0.369	99.8	96.00	96.985	**100**	98.691
$KMeansPlusPlus$	0.744	1.478	2.874	0.862	0.352	99.795	95.93	96.845	**100**	99.644
$KMeansDensity$	0.747	1.496	2.868	0.856	0.344	99.79	96.06	96.651	**100**	99.687
$KMeansVariance$	0.745	1.495	2.885	0.864	0.375	99.795	96.02	97.059	**100**	98.535
$KMeansVariance^{AvgPairWise}$	0.742	1.497	2.883	0.862	0.377	99.8	96.04	97.145	**100**	97.032
$KMeansVariance^{Version}$	0.751	1.509	2.859	0.865	0.370	99.8	96.04	97.153	**100**	99.122
$KMeansQuantiles$	0.746	1.490	2.864	0.867	0.383	99.8	96.02	97.349	**100**	98.158
$KMeansOnePass$	0.753	1.551	2.903	0.877	0.359	99.795	96.02	96.225	99.99	99.666
$KMeansSortedDistance$	0.746	1.510	2.887	0.866	0.370	99.795	96.02	97.095	**100**	98.828
$KMeansPlus^{Power}$	0.742	1.481	2.866	0.856	0.376	99.795	96.02	96.983	**100**	97.511
$KMeansPlus^{ProbPower}$	0.740	1.468	3.018	**0.852**	0.430	**99.815**	**96.11**	**100**	**100**	**100**
$KMeansPlus^{LogPower}$	**0.738**	**1.461**	3.035	0.854	0.430	**99.815**	**96.11**	**100**	**100**	**100**
$KMeansNormal^{Users}$	0.759	1.543	2.878	0.906	0.389	99.79	95.99	96.801	99.99	99.208
$KMeansNormal^{Movies}$	0.768	1.543	2.879	0.9005	0.388	99.775	95.99	96.927	99.99	99.224
$KMeansPoisson$	0.760	1.511	**2.850**	0.892	0.387	99.8	96.00	97.619	**100**	99.348
$KMeansHyperGeometric$	0.757	1.536	2.871	0.882	0.354	99.79	96.02	97.093	99.99	99.488
$KMeansUniform$	0.749	1.482	2.863	0.860	0.381	99.8	96.02	97.255	**100**	98.265
$KMeansUniform^{Version}$	0.753	1.674	2.865	0.883	0.386	99.795	95.95	97.781	99.99	99.369
$KMeansLog$	0.752	1.596	2.865	0.864	**0.336**	99.78	96.06	97.197	**100**	99.806

Table 3: Inspecting the with-in cluster similarity (TotalSim) over the test set for SML dataset, at the optimal number of clusters. The PCCDV distance measure has been used to find similarity between a user and a centroid. The best results are shown in bold font.

Centroid Selection	**TotalSim observed at diff no. of iterations (Itr)**					**Convergence**
	$Itr:2$	$Itr:4$	$Itr:6$	$Itr:8$	$Itr:10$	
$KMeans$	267.117	299.458	301.802	302.131	302.182	9
$KMeansPlus$	260.542	300.129	301.000	301.000	301.000	7
$KMeansPlusPlus$	**293.223**	297.645	300.296	301.866	302.110	10
$KMeansDensity$	205.123	271.755	300.840	302.0511	302.238	9
$KMeansVariance$	292.604	301.410	302.025	302.146	302.099	7
$KMeansVariance^{AvgPairWise}$	291.419	297.960	298.511	298.511	298.511	**6**
$KMeansVariance^{Version}$	257.866	267.093	295.433	297.552	297.581	9
$KMeansQuantiles$	283.793	296.512	301.691	305.313	305.235	9
$KMeansOnePass$	187.730	268.207	285.035	289.449	298.138	-
$KMeansSortedDistance$	267.591	**303.065**	**305.619**	**306.300**	**306.301**	8
$KMeansPlus^{Power}$	271.919	294.522	296.181	296.691	296.747	7
$KMeansPlus^{ProbPower}$	251.249	300.388	301.738	301.890	301.890	**6**
$KMeansPlus^{LogPower}$	269.863	284.024	285.250	285.809	285.835	7
$KMeanslNormal^{User}$	185.367	273.672	277.587	278.076	278.076	8
$KMeansNormal^{Movies}$	182.967	266.5146	276.478	277.290	277.483	8
$KMeansPoisson$	157.799	247.880	257.348	258.855	258.855	8
$KMeansHyperGeometric$	193.663	230.730	279.440	280.645	280.799	8
$KMeansUniform$	257.681	301.872	303.750	303.795	303.795	**6**
$KMeansUniform^{Version}$	166.200	205.432	210.655	213.201	214.633	10
$KMeansLog$	185.120	229.862	274.998	275.986	276.055	10

Table 4: Inspecting the with-in cluster similarity (TotalSim) over the test set for FT dataset, at the optimal number of clusters. The PCCDV distance measure has been used to find similarity between a user and a centroid. The best results are shown in bold font.

Centroid Selection	**TotalSim observed at diff no. of iterations (Itr)**					**Convergence**
	$Itr:2$	$Itr:4$	$Itr:6$	$Itr:8$	$Itr:10$	
$KMeans$	313.041	395.707	400.298	404.941	406.928	-
$KMeansPlus$	356.770	**407.120**	410.212	411.229	414.259	-
$KMeansPlusPlus$	324.027	406.483	**412.789**	**413.841**	415.836	-
$KMeansDensity$	226.826	401.112	411.904	413.811	**415.861**	-
$KMeansVariance$	316.914	398.853	408.602	408.960	408.711	**7**
$KMeansVariance^{AvgPairWise}$	349.549	392.253	402.962	404.041	408.935	-
$KMeansVariance^{Version}$	224.515	295.851	337.295	376.378	376.989	9
$KMeansQuantile$	314.642	389.163	402.932	404.686	406.593	-
$KMeansOnePass$	309.879	376.668	384.373	387.501	387.350	9
$KMeansSortedDistance$	**383.295**	406.755	410.389	412.394	412.464	9
$KMeansPlus^{Power}$	125.189	144.483	149.578	149.867	151.913	-
$KMeansPlus^{ProbPower}$	377.911	400.511	404.096	407.241	407.624	9
$KMeansPlus^{LogPower}$	380.122	397.606	402.108	403.23	404.582	-
$KMeansNormal^{Users}$	307.528	401.479	411.113	407.979	410.185	-
$KMeansNormal^{Movies}$	307.528	401.479	411.113	407.979	410.185	-
$KMeansPoisson$	308.543	371.509	380.214	384.980	387.721	-
$KMeansHyperGeometric$	252.912	389.895	401.278	403.584	404.097	-
$KMeansUniform$	353.010	397.448	405.650	406.441	406.992	8
$KMeansUniform^{Version}$	236.940	317.536	332.904	335.627	331.613	-
$KMeansLog$	152.128	219.401	351.384	385.705	390.896	-

Table 5: Inspecting the with-in cluster similarity (TotalSim) over the test set for BC dataset, at the optimal number of clusters. The PCCwithoutDefault distance measure has been used to find similarity between a user and a centroid. The best results are shown in bold font.

Centroid Selection	**TotalSim observed at diff no. of iterations (Itr)**					**Convergence**
	$Itr : 2$	$Itr : 4$	$Itr : 6$	$Itr : 8$	$Itr : 10$	
$KMeans$	231.519	234.181	235.243	235.925	236.091	10
$KMeansPlus$	229.053	233.641	238.494	239.950	241.600	-
$KMeansPlusPlus$	87.111	157.772	173.367	179.237	181.057	-
$KMeansDensity$	32.8724	182.494	210.227	217.040	219.857	-
$KMeansVariance$	227.526	237.565	239.301	239.537	239.685	7
$KMeansVariance^{AvgPairWise}$	229.791	246.737	249.650	249.877	249.934	7
$KMeansVariance^{Version}$	210.906	199.745	202.138	205.228	203.056	-
$KMeansQuantiles$	188.186	234.952	238.232	239.040	239.306	9
$KMeansOnePass$	32.8724	109.897	119.613	122.663	123.465	10
$KMeansSortedDistance$	240.633	248.710	249.379	249.573	249.724	6
$KMeansPlus^{Power}$	194.522	207.402	212.630	215.923	217.476	-
$KMeansPlus^{ProbPower}$	256.266	256.973	257.360	257.435	257.452	6
$KMeansPlus^{LogPower}$	**410.566**	**410.460**	**409.945**	**410.035**	**409.890**	**3**
$KMeanslNormal^{Users}$	38.6327	287.136	291.095	293.910	294.200	9
$KMeansNormal^{Movies}$	39.639	287.334	288.656	289.197	289.894	8
$KMeansPoisson$	46.238	159.231	186.262	190.143	192.605	-
$KMeansHyperGeometric$	47.463	64.285	69.596	75.5416	78.7656	-
$KMeansUniform$	207.633	223.082	226.521	227.566	227.943	8
$KMeansUniform^{Version}$	32.8724	180.948	210.089	213.698	228.700	-
$KMeansLog$	67.0831	94.1721	99.336	110.905	110.581	9

Table 6: Inspecting the with-in cluster similarity (TotalSim) over the test set for ML dataset, at the optimal number of clusters. The PCCwithoutDefault distance measure has been used to find similarity between a user and a centroid. The best results are shown in bold font.

Centroid Selection	**TotalSim observed at diff no. of iterations (Itr)**					**Convergence**
	$Itr : 2$	$Itr : 4$	$Itr : 6$	$Itr : 8$	$Itr : 10$	
$KMeans$	1304.994	1341.641	1365.691	1381.625	1364.263	-
$KMeansPlus$	1360.072	1368.232	1396.591	1400.026	1405.320	-
$KMeansPlusPlus$	1307.144	1342.322	1370.742	1385.479	1397.060	-
$KMeansDensity$	905.214	1168.651	1193.193	1227.377	1235.821	-
$KMeansVariance$	1320.235	1349.232	1359.784	1380.235	1389.992	-
$KMeansVariance^{AvgPairWise}$	1308.959	1342.484	1366.106	1380.664	1388.922	-
$KMeansVariance^{Version}$	1401.208	1434.745	1449.818	1453.075	1455.25	10
$KMeansQuantiles$	1306.796	1324.766	1357.847	1351.772	1373.844	-
$KMeansOnePass$	914.601	1114.165	1110.135	1103.081	1118.694	-
$KMeansSortedDistance$	1311.003	1335.511	1333.004	1356.451	1370.713	-
$KMeansPlus^{Power}$	1262.301	1305.480	1331.022	1347.630	1358.320	-
$KMeansPlus^{ProbPower}$	1325.611	1358.311	1351.131	1370.375	1399.151	-
$KMeansPlus^{LogPower}$	**1529.412**	**1535.811**	**1540.391**	**1545.852**	**1545.982**	**9**
$KMeanslNormal^{Users}$	925.143	1069.471	1126.808	1139.001	1156.952	-
$KMeansNormal^{Movies}$	925.145	1077.022	1116.182	1135.342	1144.851	-
$KMeansPoisson$	937.631	1048.001	1080.415	1096.273	1107.265	-
$KMeansHyperGeometric$	937.583	1060.014	1101.184	1111.854	1115.794	-
$KMeansUniform$	1312.881	1326.236	1339.532	1340.093	1371.143	-
$KMeansUniform^{Version}$	905.212	1094.532	1170.811	1193.961	1207.621	-
$KMeansLog$	929.761	959.771	1003.041	1011.712	1018.522	-

Table 7: Inspecting the with-in cluster similarity (TotalSim) over the test set for FM dataset, at the optimal number of clusters. The PCCwithoutDefault distance measure has been used to find similarity between a user and a centroid. The best results are shown in bold font.

Centroid Selection	**TotalSim observed at diff no. of iterations (Itr)**					**Convergence**
	$Itr:2$	$Itr:4$	$Itr:6$	$Itr:8$	$Itr:10$	
$KMeans$	59.893	65.662	69.141	69.729	70.097	10
$KMeansPlus$	60.485	66.189	69.619	70.313	70.613	9
$KMeansPlusPlus$	7.068	39.916	59.679	63.718	65.433	-
$KMeansDensity$	1.551	53.474	64.272	68.300	70.292	-
$KMeansVariance$	61.947	68.271	69.223	69.563	69.820	8
$KMeansVariance^{AvgPairWise}$	61.018	68.445	69.138	69.622	69.758	**7**
$KMeansVariance^{Version}$	51.147	62.043	66.461	70.069	70.997	9
$KMeansQuantiles$	61.596	68.064	69.385	69.867	70.151	10
$KMeansOnePass$	1.551	40.454	60.977	65.337	66.696	10
$KMeansSortedDistance$	60.657	67.111	69.498	70.153	70.247	9
$KMeansPlus^{Power}$	61.426	68.235	69.084	69.481	69.761	**7**
$KMeansPlus^{ProbPower}$	**69.757**	**71.301**	**71.715**	**72.072**	**72.186**	**7**
$KMeansPlus^{LogPower}$	64.660	68.801	69.639	70.072	70.151	8
$KMeanslNormal^{Users}$	1.551	61.30	64.018	65.085	65.741	8
$KMeansNormal^{Movies}$	2.952	62.177	64.493	64.962	65.274	10
$KMeansPoisson$	2.670	45.585	59.234	61.224	62.361	10
$KMeansHyperGeometric$	4.833	46.836	57.594	59.280	59.961	9
$KMeansUniform$	60.082	65.602	69.883	71.818	71.981	9
$KMeansUniform^{Version}$	1.551	56.154	66.184	67.065	67.527	9
$KMeansLog$	3.893	56.921	62.713	66.001	67.029	10

12 ($KMeansPlus^{ProbPower}$) shows highest within-cluster similarity for LastFM dataset shown in Table 7.

5.3. Performance comparison in terms of cluster convergence

We claimed that initial centroid selection affects the convergence rate of the clusters and consequently recommendation process as well. To prove our statement we present the convergence rate (the difference between with-in cluster similarity become negligible with increasing number of iterations) of clusters in Table 3, 4, 5, 6, and 7 for SML, FT, BC, ML and FM datasets respectively.

We observe for SML and FT datasets, Algorithm 5 $KMeansVariance$ and Algorithm 10 $KMeansSortedDistance$ provide speedy convergence as compared to all other approaches as shown graphically in Figure 3 and 4. While for BC, ML and FT datasets Algorithm 13 ($KMeansPlus^{LogPower}$) and Algorithm 12 ($KMeansPlus^{ProbPower}$ show speedy covergence as shown in Figure 5, 6, and 7.

5.4. Performance comparison in terms of cluster building time

We proposed some centroid selection algorithms for the situation where we dont have enough training time. Figure B.11 presents the comparison of different centroid selection approaches, in term of time to build the clusters, and shows that centroid selection approaches using data distribution technique takes the minimum time for cluster building.

5.5. Performance comparison in terms of recommendation coverage

Beyond accuracy measures we have also compared the performance of different centroid selection approaches for a non-accuracy measure that is coverage. To increase the recommendation coverage we have proposed Algorithm 12 (i.e. $KMeansPlus^{ProbPower}$) and Algorithm 13 (i.e $KMeansPlus^{LogPower}$) . Results in Figure B.12 shows that both of aforementioned algorithms outperform all of the rest algorithms for all five datasets.

5.6. Performance comparison under cold start problems

5.6.1. New user cold-start

We performed experiments to investigate the performance of centroid selection approaches under new user cold-start scenario. For that we randomly selected 100 users, and proceeded the experiment by keeping their number of ratings in the training set to 2, 10, and 20. The results are displayed in Table 10 and Table 11, and the corresponding MAE is represented by MAE 2, MAE 10, and MAE 20. The results over all 5 datasets show that the traditional approach is highly affected by these scenarios. The proposed approaches improve the performance and provide better recommendation. For comparison purpose, some best approaches along with conventional k-means are shown in Table 10 and Table 11.

5.6.2. New item cold start

We performed experiments to investigate the performance of centroid selection approaches under new item

cold-start scenario. For that we randomly selected 100 items, and proceeded the experiment by keeping their number of ratings in the training set to 2, 10, and 20. The results are displayed in Table 12 and Table 13, and the corresponding MAE is represented by MAE 2, MAE 10, and MAE 20. Table 12 and Table 13 show that the performance of the conventional k-means suffer under these scenarios, for all 5 datasets. The proposed approaches improve the performance and provide better recommendation. For comparison purpose, some best approaches along with conventional k-means are shown in Table 12 and Table 13.

5.7. Performance comparison under long tail problems

We produced artificial tail by randomly selecting 20% of items in the tail, for testing the performance of centroid selection approaches under long tail scenario. We kept the ratings in the tail to 2, 10, and 20 and the corresponding MAE—represented by MAE 2, MAE 10, and MAE 20—is shown in Table 14 and Table 15. It is observed that performance under long tail scenario is the same as in new item cold-start condition.

5.8. Interpretation with traditional k-means

The rationale behind proposing various algorithms is to provide better recommendations under different conditions (different datasets, matrices, training time, etc). On experimentation over SML, ML, and FT dataset, we found that many of our proposed approaches outperform traditional k-means in terms of MAE, while Algorithm 13 (i.e. $KMeansPlus^{LogPower}$) and Algorithm 12 (i.e. $KmeansPlus^{ProbPower}$) perform the best.

For BC and FM datasets we observe that most of centroid selection approaches perform better than $KMeans$ but approaches based on data distribution performs better than all others. Meanwhile Algorithm 4 ($KMeansDensity$) gives comparable result not only for ML dataset but also for FM dataset.

The % decrease in MAE (for the best identified algorithm) over the baseline approach (Algorithm 1) is found to be (1) 0.94% in the case of Algorithm 13 with $p < 0.1$ (p-value in the case of pair-t test) for the SML dataset; (2) 2.14% in the case of Algorithm 13 with $p < 0.01$ (p-value in the case of pair-t test) for the FT dataset; (3) 1.21% in the case of Algorithm 16 with $p < 0.05$ (p-value in the case of pair-t test) for the BC dataset; (4)1.27% in the case of Algorithm 12 with $p < 0.05$ (p-value in the case of pair-t test) for the ML dataset; and (4) 10.63% in the case of Algorithm 20 with $p < 0.0001$ (p-value in the case of pair-t test) for the FM dataset. These values show that the results are statistically significant (using $p < 0.05$ as cut-off point), especially for FM, FT, and ML datasets.

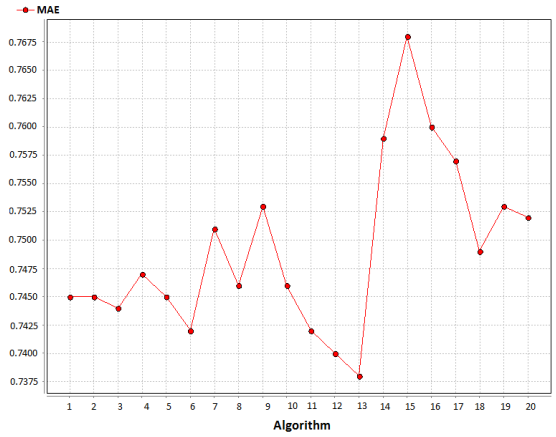
In our work various centroid selection approaches significantly improves cluster quality as shown in Figure 3, 4, 5, 6, and 7. We also observe that the recommendation coverage of Algorithm 12 ($KMeansPlus^{ProbPower}$) and Algorithm 13 ($KMeansPlus^{LogPower}$) is better than all others, for all five datasets as shown in Figure B.12. Figure B.11 shows that different centroid selection approaches based on data distributions provide significant improvements in term of cluster time complexity, so we can use these approaches where we have less time to train our dataset.

Based on the experimental results, we can highlight the following key points: (1) The investigated seed selections approaches exhibit much superior accuracy and performance compared to traditional seed selection approach—random selection. However, the results of various seed selection methods are dataset dependent and no approach is a panacea. Depending on the dataset properties (e.g. distribution, sparsity, etc.) one approach might give very good results—in terms of MAE and coverage—for one approach and give average results for other; (2) $KMeansPlus^{ProbPower}$ and $KMeansPlus^{LogPower}$ gave consistently better results than other approaches. The reason is these approaches take the concept of leaders and followers and in addition combine this concept with K-Means++'s concept; (3) For relatively smaller and very sparse and imbalanced datasets (sparsity > 99%), combining the concepts of power users and KMeans++ gives better performance. The reason is the same as discussed in 3.1 (that most users tend to follow leader's opinions in online communities) ; (4) The approaches based on data distribution can lead to significant saving in time and gives much better results for larger datasets (e.g. BC dataset), (5) For binary class datasets (e.g. FM dataset), the approaches based on data distribution and density gives better results; (6) Different seed selection approaches can be selected depending different priorities and circumstances and—accuracy requirement, coverage requirement, and time and frequency of running the off-line computation.

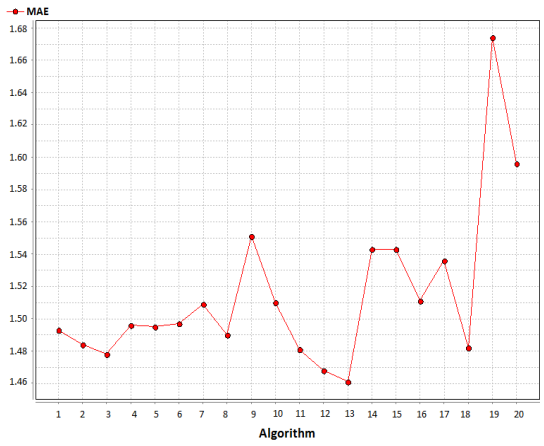
5.9. Centroid Selection in Fuzzy C-means Clustering (FCM)

Hard clustering such as k-means assigns each user to only one cluster though in reality user may have diverse opinions; FCM is used in literature based on same consideration. FCM algorithm behaves in the same fashion as k-means because basically it is similar in structure to k-means algorithm. The additional feature in FCM is that it allows user to be part of multiple clusters based on their interest.Therefore, this approach can produce better results but just multiple assignment is not enough to make better recommendations for all type of datasets. As cluster quality in FCM clustering also depends on initial centroid selection, randomly chosen centroids mostly end up with inaccurate results. One of the major drawback of FCM is that it produces inaccurate recommendation especially in large datasets [62] , because the randomly chosen initial centroids are not well separated and irrelevant data elements can be assigned to clusters which may overwhelm the rating prediction.

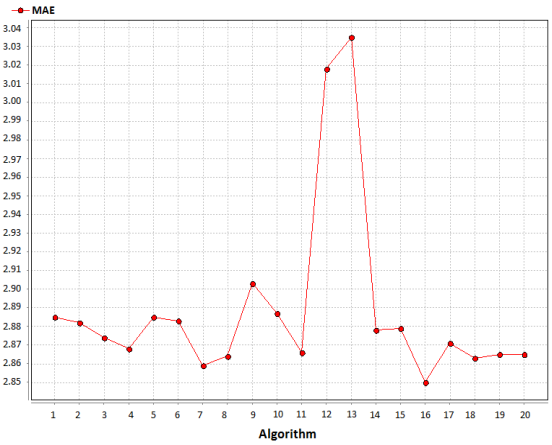
We applied the proposed centroid selection approaches on FCM and calculated the results by keeping rest of the



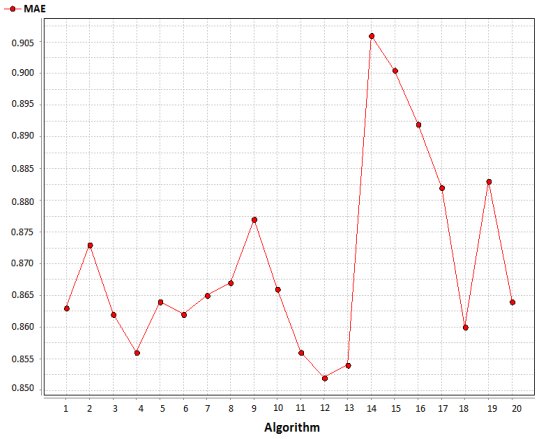
(a) MAE over SML dataset



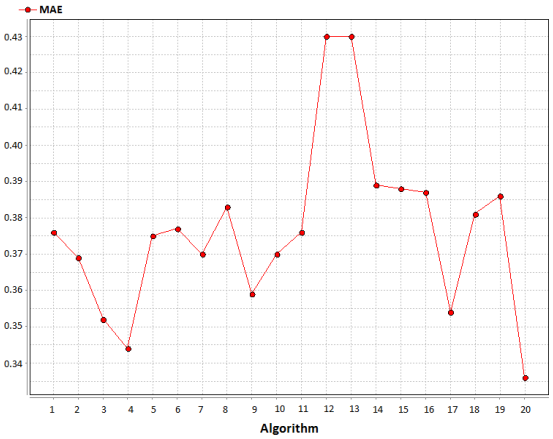
(b) MAE over FT dataset



(c) MAE over BC dataset



(d) MAE over ML dataset



(e) MAE over FM dataset

Figure 2: Inspecting the performance of different centroid selection approaches by measuring MAE over FT, SML, BC, ML and FM datasets. The horizontal axis depicts different algorithms while the vertical axis represents the corresponding MAE. The results show that Algorithm 12 ($KmeansPlus^{ProbPower}$) and Algorithm 13 ($KMeansPlus^{LogPower}$)) give best results for SML, FT, and ML datasets. Results of BC dataset represent that Algorithm 16,($KMeansPoisson$) gives minimum MAE while for FM dataset Algorithm 20 ($KmeansLog$) provides best results.

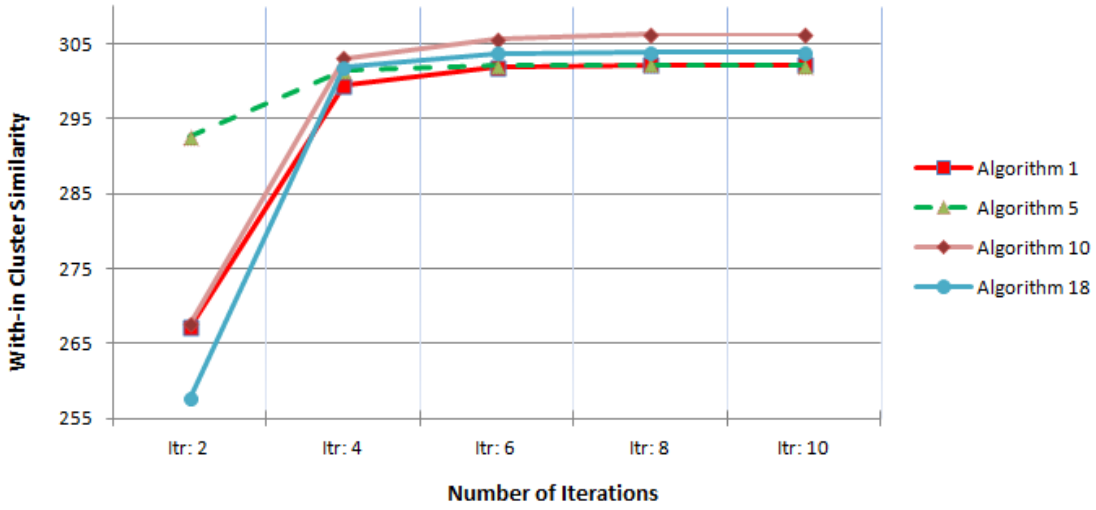


Figure 3: Comparing the cluster similarity of few proposed centroid selection approaches with conventional KMeans over SML dataset.

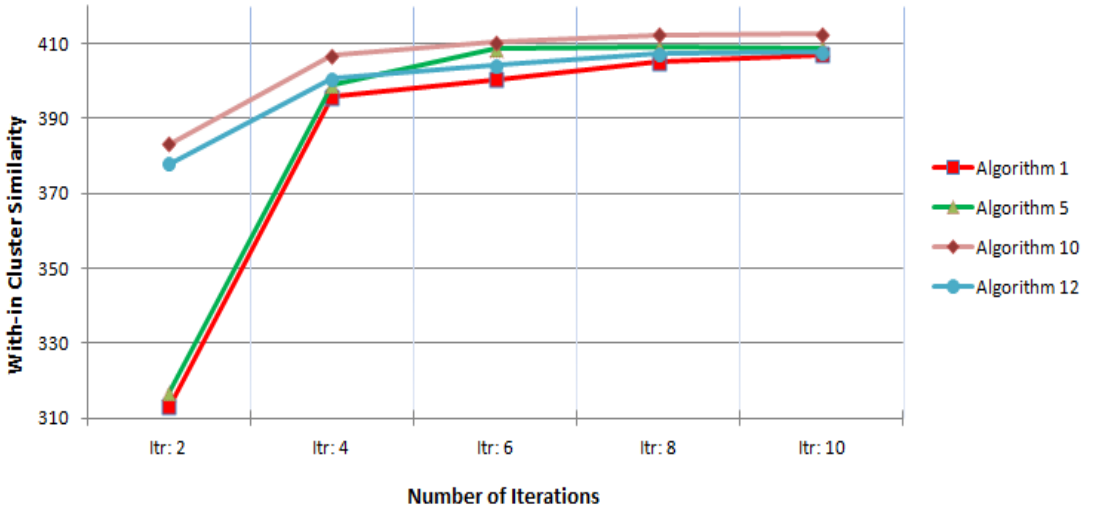


Figure 4: Comparing the cluster similarity of few proposed centroid selection approaches with conventional KMeans over FT dataset.

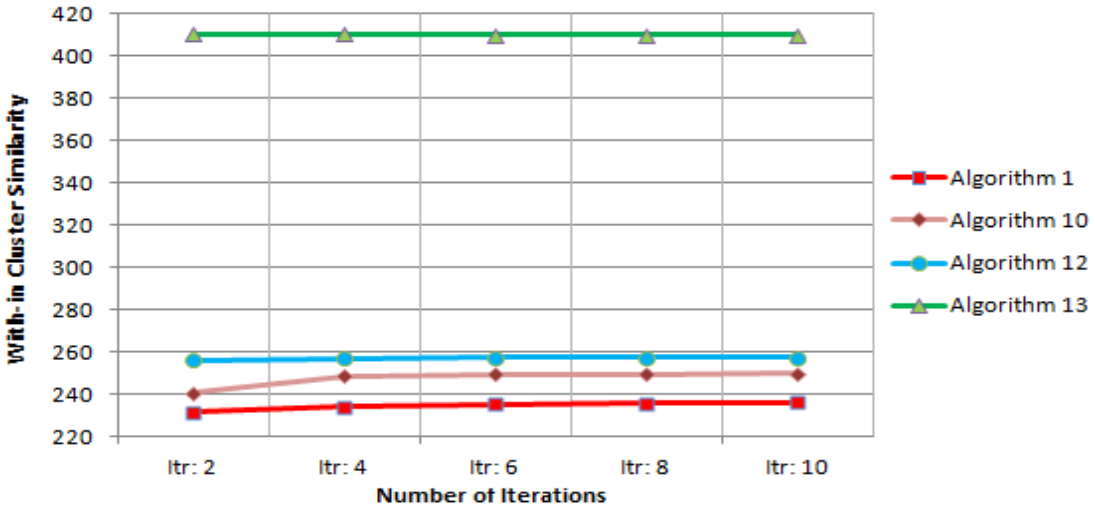


Figure 5: Comparing the cluster similarity of few proposed centroid selection approaches with conventional KMeans over BC dataset.

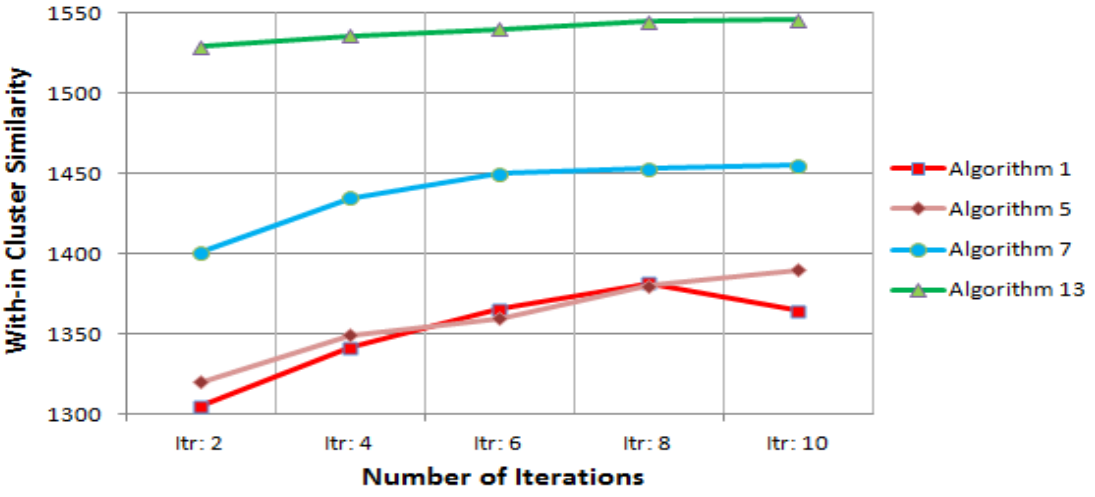


Figure 6: Comparing the cluster similarity of few proposed centroid selection approaches with conventional KMeans over ML dataset.

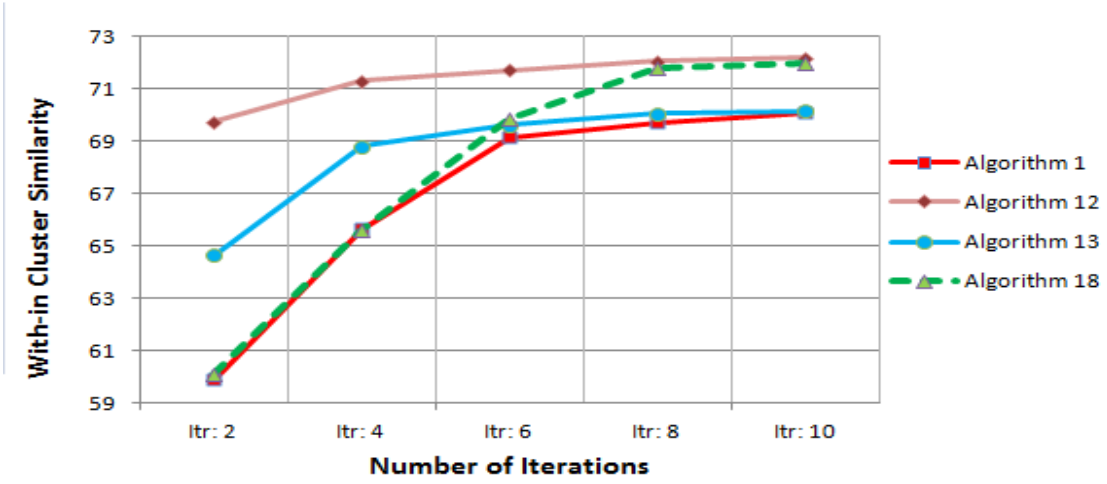


Figure 7: Comparing the cluster similarity of few proposed centroid selection approaches with conventional KMeans over FM dataset.

algorithm intact, and compared the results with traditional FCM. Table 8 shows that traditional FCM algorithm shows almost the same results as k-means over small dataset like FM. But its MAE keeps on increasing with increase in the size of dataset. Our proposed approaches improves the performance of FCM in terms of accuracy and coverage as given in Table 8. The reason is that better initial centroid selection improves the quality of clusters which ultimately provide better recommendation. As we know that FCM assigns a data element to multiple clusters hence its coverage is better than traditional k-means, and proposed solution improves it even more, as given in Table 8.

5.10. Centroid Selection in Expectation-maximization (EM)

We applied the proposed centroid selection approaches on EM and calculated the results by keeping rest of the algorithm intact, and compared the results of proposed solution with conventional EM. Table 9 shows the results of EM algorithm along with all proposed centroid selection approaches implemented on EM. Table 9 shows that proposed approaches improves the result of traditional EM and provides almost the same results as FCM presented in Table 8. The reason is that initial parameter is selected effectively with proposed centroid selection appraches which consequently improves the quality of clusters and provide better recommendation.

6. Conclusion and Future Work

The main claim of this work is that centroids of k-means clustering algorithms, if effectively selected prior to partition the recommender system dataset into different clusters, can provide potential benefits ranging from cost saving to performance enhancement. This work, presents a comparative study of the centroid selection approaches in k-means-based recommender system and their subsequent impact upon the accuracy and cost has been investigated. A second analysis of centroid selection approaches is performed over Fuzzy C-means and Expectation-maximization algorithm, to investigate the performance of proposed solution on related clustering algorithm. The empirical study has shown that rather than using the traditional approach to select the centroids in centroid model-based (k-mean/FCM/EM) recommender; which have heavily been used in the literature, robust and advanced approaches can give considerable performance benefits.

A limitation of k-means algorithm is that it highly depends on k number of clusters and k must be predefined. Developing some statistical methods to compute the value of k, depending on the data distribution, is suggested for future research. An alternative approach for seed selection is to exploit more than one user as the initial centroid, which might accelerate the convergence rate of the k-means clustering algorithm.

Appendix A. Learning the Optimal System Parameters

Appendix A.1. Optimal number of clusters

We changed the number of clusters from 1 to 200 for tuning optimal number of clusters for k-means clustering of SML and FT datasets, and measured the corresponding MAE. Figure 8(a) depicts that for SML dataset, the MAE keeps on decreasing with an increase in number of clusters, however after 130 clusters, this behavior is not significant. Similarly when tuned on FT dataset, the MAE shows minimum value at 130 and 190 clusters as displayed in Figure 8(a). We choose 130 clusters to be optimal for both the datasets to be used for subsequent analysis. Meanwhile to

Table 8: Inspecting the performance of different centroid selection approaches on Fuzzy C-means (FCM) clustering, in terms of MAE and coverage, over the test set. In Algorithm notations, $KMeans$ is replaced by $FCMeans$ as proposed approaches are implemented using FCM clustering. The best results are shown in bold font.

	MAE					**Coverage**				
Centroid Selection	**SML**	**FT**	**BC**	**ML**	**FM**	**SML**	**FT**	**BC**	**ML**	**FM**
$FCMeans$	0.792	1.739	3.199	1.381	0.356	99.815	96.24	98.197	**100**	99.079
$FCMeansPlus$	0.776	1.674	3.185	1.375	0.338	99.915	96.21	97.684	**100**	99.581
$FCMeansPlusPlus$	0.775	1.668	3.178	1.363	0.321	99.825	96.14	97.444	**100**	99.982
$FCMeansDensity$	0.778	1.716	3.163	1.357	0.315	99.827	96.27	97.352	**100**	99.978
$FCMeansVariance$	0.777	1.715	3.189	1.366	0.344	99.825	96.24	97.758	**100**	99.841
$FCMeansVariance^{AvgPairWise}$	0.773	1.717	3.187	1.363	0.346	99.916	96.25	97.846	**100**	97.921
$FCMeansVariance^{Version}$	0.780	1.729	3.156	1.366	0.341	99.905	96.23	97.854	**100**	99.131
$FCMeansQuantiles$	0.777	1.710	3.167	1.368	0.352	99.905	96.24	97.983	**100**	98.161
$FCMeansOnePass$	0.781	1.731	3.109	1.376	0.330	99.831	96.25	96.927	99.99	99.891
$FCMeansSortedDistance$	0.777	1.710	3.184	1.367	0.341	99.823	96.27	97.798	**100**	99.656
$FCMeansPlus^{Power}$	0.773	1.671	3.165	1.357	0.345	99.823	96.29	97.681	**100**	98.498
$FCMeansPlus^{ProbPower}$	0.772	**1.658**	3.171	1.353	0.390	**99.985**	**97.45**	**100**	**100**	**100**
$FCMeansPlus^{LogPower}$	**0.767**	1.662	3.198	**1.245**	0.361	**99.985**	97.11	**100**	**100**	**100**
$FCMeansNormal^{Users}$	0.781	1.741	3.127	1.303	0.348	99.825	96.21	97.499	99.99	99.718
$FCMeansNormal^{Movies}$	0.797	1.741	3.128	1.311	0.347	99.804	96.21	97.617	99.99	99.618
$FCMeansPoisson$	0.791	1.722	**3.001**	1.289	0.346	99.915	96.23	98.321	**100**	99.822
$FCMeansHyperGeometric$	0.786	1.732	3.124	1.279	0.322	99.818	96.23	98.012	99.99	99.918
$FCMeansUniform$	0.777	1.679	3.115	1.271	0.342	99.915	96.23	98.015	**100**	99.121
$FCMeansUniform^{Version}$	0.781	1.722	3.119	1.291	0.345	99.815	96.18	98.439	99.99	99.786
$FCMeansLog$	0.781	1.738	3.121	1.265	**0.301**	99.812	96.28	97.988	**100**	99.99

Table 9: Inspecting the performance of different centroid selection approaches on Expectation-maximization (EM) algorithm, in terms of MAE and coverage, over the test set. In Algorithm notations, $KMeans$ is replaced by EM as proposed approaches are implemented using EM algorithm. The best results are shown in bold font.

	MAE					**Coverage**				
Centroid Selection	**SML**	**FT**	**BC**	**ML**	**FM**	**SML**	**FT**	**BC**	**ML**	**FM**
EM	0.799	1.741	3.189	1.383	0.357	99.75	96.25	98.255	**100**	99.015
$EMPlus$	0.774	1.681	3.175	1.377	0.337	99.25	96.15	97.685	**100**	99.576
$EMPlusPlus$	0.785	1.672	3.168	1.365	0.331	99.825	96.75	97.454	**100**	99.895
$EMDensity$	0.787	1.701	3.158	1.359	0.319	99.755	96.95	97.359	**100**	99.585
$EMVariance$	0.771	1.709	3.179	1.366	0.341	99.675	96.75	97.858	**100**	99.546
$EMVariance^{AvgPairWise}$	0.769	1.699	3.181	1.365	0.345	99.915	96.85	97.546	**100**	97.895
$EMVariance^{Version}$	0.783	1.704	3.149	1.366	0.343	99.925	97.25	97.875	**100**	99.155
$EMQuantiles$	0.771	1.722	3.158	1.371	0.357	99.925	97.35	97.975	**100**	98.165
$EMOnePass$	0.779	1.727	3.121	1.373	0.328	99.85	96.25	96.985	99.95	99.891
$EMSortedDistance$	0.775	1.713	3.177	1.361	0.345	99.815	96.275	97.795	**100**	99.575
$EMPlus^{Power}$	0.778	1.673	3.156	1.363	0.343	99.815	96.295	97.665	**100**	98.658
$EMPlus^{ProbPower}$	**0.763**	**1.656**	3.219	1.356	0.368	**99.985**	**97.75**	**100**	**100**	**100**
$EMPlus^{LogPower}$	0.769	1.668	3.178	**1.249**	0.368	**99.985**	97.45	**100**	**100**	**100**
$EMNormal^{Users}$	0.779	1.731	3.135	1.308	0.347	99.75	96.75	97.45	99.95	99.875
$EMNormal^{Movies}$	0.794	1.732	3.119	1.312	0.351	99.815	96.85	97.65	99.95	99.659
$EMPoisson$	0.793	1.725	3.117	1.291	0.343	99.905	96.85	98.555	**100**	99.755
$EMHyperGeometric$	0.781	1.737	3.123	1.282	0.327	99.815	96.25	98.175	99.95	99.918
$EMUniform$	0.769	1.699	**3.114**	1.273	0.332	99.925	96.25	98.215	**100**	99.125
$EMUniform^{Version}$	0.791	1.735	3.121	1.289	0.347	99.825	96.85	98.435	99.95	99.855
$EMLog$	0.789	1.739	3.118	1.272	**0.313**	99.825	96.75	97.857	**100**	99.99

Table 10: Inspecting the performance of different centroid selection approaches under **new-user cold start** in terms of MAE, over SML, FT and BC datasets. The best results are shown in bold font.

Centroid Selection	SML			FT			BC		
	MAE2	**MAE10**	**MAE20**	**MAE2**	**MAE10**	**MAE20**	**MAE2**	**MAE10**	**MAE20**
$KMeans$	1.107	0.948	0.892	1.8030	1.756	1.701	3.459	3.339	3.257
$KMeansDensity$	1.056	0.891	0.852	1.765	1.751	1.675	3.422	3.301	3.211
$KMeansSortedDistance$	1.045	0.901	0.843	1.754	1.749	1.651	3.432	3.289	3.205
$KMeansPlus^{ProbPower}$	1.021	0.872	**0.811**	**1.721**	1.745	1.632	**3.401**	3.251	3.197
$KMeansPlus^{LogPower}$	**1.011**	**0.869**	**0.811**	1.729	**1.743**	**1.611**	3.411	**3.233**	**3.181**

Table 11: Inspecting the performance of different centroid selection approaches under **new-user cold start** in terms of MAE, over ML and FM datasets. The best results are shown in bold font.

Centroid Selection	ML			FM		
	MAE2	**MAE10**	**MAE20**	**MAE2**	**MAE10**	**MAE20**
$KMeans$	1.215	1.149	1.085	0.586	0.535	0.521
$KMeansDensity$	1.149	1.091	1.052	0.545	0.521	0.491
$KMeansSortedDistance$	1.165	1.101	**1.031**	0.534	0.517	0.473
$KMeansPlus^{ProbPower}$	1.151	**1.085**	1.059	**0.521**	**0.505**	0.454
$KMeansPlus^{LogPower}$	**1.144**	1.093	1.034	0.522	0.511	**0.438**

Table 12: Inspecting the performance of different centroid selection approaches under **new-item cold start** in terms of MAE, over SML, FT and BC datasets. The best results are shown in bold font.

Centroid Selection	SML			FT			BC		
	MAE2	**MAE10**	**MAE20**	**MAE2**	**MAE10**	**MAE20**	**MAE2**	**MAE10**	**MAE20**
$KMeans$	0.981	0.911	0.872	1.785	1.721	1.685	3.397	3.365	3.285
$KMeansDensity$	0.965	0.878	0.842	1.723	1.701	1.645	3.345	3.315	3.254
$KMeansSortedDistance$	0.941	0.893	**0.815**	**1.712**	1.699	1.634	3.321	3.309	3.215
$KMeansPlus^{ProbPower}$	**0.912**	**0.858**	0.832	1.732	1.674	**1.610**	3.319	3.299	3.205
$KMeansPlus^{LogPower}$	0.921	0.864	0.821	1.722	**1.669**	1.611	**3.305**	**3.284**	**3.201**

Table 13: Inspecting the performance of different centroid selection approaches under **new-item cold start** in terms of MAE, over ML and FM datasets. The best results are shown in bold font.

Centroid Selection	ML			FM		
	MAE2	**MAE10**	**MAE20**	**MAE2**	**MAE10**	**MAE20**
$KMeans$	1.196	1.165	1.088	0.575	0.529	0.481
$KMeansDensity$	1.151	1.119	1.057	0.545	0.491	0.465
$KMeansSortedDistance$	1.145	1.104	1.043	0.562	0.487	0.457
$KMeansPlus^{ProbPower}$	1.125	**1.087**	**1.011**	0.521	**0.475**	**0.421**
$KMeansPlus^{LogPower}$	**1.121**	1.091	1.019	**0.512**	0.476	0.429

Table 14: Inspecting the performance of different centroid selection approaches under **long tail** in terms of MAE, over SML, FT and BC datasets. The best results are shown in bold font.

Centroid Selection	SML			FT			BC		
	MAE2	**MAE10**	**MAE20**	**MAE2**	**MAE10**	**MAE20**	**MAE2**	**MAE10**	**MAE20**
$KMeans$	0.985	0.927	0.881	1.791	1.725	1.681	3.387	3.375	3.281
$KMeansDensity$	0.963	0.865	0.845	1.728	1.704	1.643	3.343	3.319	3.261
$KMeansSortedDistance$	0.942	0.876	0.825	**1.715**	1.689	1.644	**3.311**	3.306	3.224
$KMeansPlus^{ProbPower}$	0.919	**0.835**	**0.822**	1.722	1.676	**1.617**	3.316	3.289	3.211
$KMeansPlus^{LogPower}$	**0.924**	0.844	0.826	1.725	**1.671**	1.620	3.309	**3.282**	**3.209**

Table 15: Inspecting the performance of different centroid selection approaches under **long tail** in terms of MAE, over ML and FM datasets. The best results are shown in bold font.

Centroid Selection	ML			FM		
	MAE2	**MAE10**	**MAE20**	**MAE2**	**MAE10**	**MAE20**
$KMeans$	1.198	1.163	1.085	0.577	0.523	0.467
$KMeansDensity$	1.161	1.127	1.056	0.556	0.471	0.445
$KMeansSortedDistance$	1.149	1.112	1.063	0.551	0.485	0.432
$KMeansPlus^{ProbPower}$	1.126	1.088	1.024	0.528	0.463	**0.413**
$KMeansPlus^{LogPower}$	**1.121**	**1.071**	**1.018**	**0.521**	**0.461**	0.429

find the optimal number of clusters for k-means clustering, to be used for BC and ML datasets, we iterate from 100 to 400 clusters and measures the corresponding MAE. The MAE shows best results at 150 clusters but after that it keep on increasing with an increase in number of clusters for both the datasets, as shown in Figure 8(a). For this reason we choose 150 clusters, as optimal value for BC and ML datasets. While tuning optimal number of clusters for FM dataset, we started from 20 clusters and found that MAE keep on decreasing till 80 clusters but after that it increasing. So, we choose 80 clusters to be optimal for FM dataset, as shown in Figure 8(a).

Appendix A.2. Optimal number of neighbours for the CCF

To find the optimal number of neighbours to be used in CCF algorithms, for SML, FT and FM datasets, we iterate from 10 to 100 neighbours and measured the corresponding MAE. The MAE decreases significantly till 60 neighbours when tuned on FT dataset set, as shown in Figure 8(b). While in the case of SML dataset, the MAE shows best result at 30 neighbours but after that keeps increasing with increase in number of neighbours and for FM dataset MAE keeps decreasing till 50, but shows no variation afterwards, as shown in Figure 8(b). So we choose 60 neighbours for SML, 30 for FT, and 50 for FM dataset as optimal values to be used for subsequent analysis. Also we changed the number of neighbours from 10 to 160 for tuning optimal number of neighbours for k-means clustering of BC and ML dataset, and measured the corresponding MAE. Figure 8(b) depicts that the MAE keep on decreasing with an increase in number of neighbours till 100 neighbours, but after that it increases — for both the dataset. For this reason we choose 100 neighbours, as optimal neighbourhood size for BC and ML datasets.

Appendix A.3. Optimal number of iterations

To find optimal number of iterations for k-means clustering, we change the iterations from 1 to 8 (keeping other parameters fixed). Figure 8(c) shows the variations in MAE when tuned on SML dataset. Though at iteration = 6, the MAE shows minimum value but we observe that after 4 iterations, decrease in MAE is insignificant. Therefore we choose optimal number of iterations to be 4, to keep a good balance between performance and computational requirements. For FT dataset we observe that MAE shows significant improvement at iteration = 2, after that MAE keep on increasing with an increase in number of iterations. So we choose optimal number of iterations to be 2 for FT dataset. Similarly when tuned on BC, ML and FM datasets, the MAE decreases significantly till 5 iterations and shows minimum value at 5 iterations as shown in Figure 8(c). For this reason we choose 5 iterations, as optimal value for BC, ML, and FM datasets.

Appendix B. Result of Experiments on different Centroid Selection Approaches

Figures in this section show results of different centroid selection approaches over optimal parameter tuned for this work. Brief summary of the results is given in figure description.

References

[1] Adomavicius, G., & Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *Knowledge and Data Engineering, IEEE Transactions on*, *17*, 734–749.

[2] Amorim, R. C. D. (2011). Learning feature weights for k-means clustering using the minkowski metric.

[3] Arai, K., & Barakbah, A. R. (2007). *Hierarchical K-means: an algorithm for centroids initialization for K-means*. Technical Report Faculty of Science and Engineering, Saga University.

[4] Arthur, D., & Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* SODA '07 (pp. 1027–1035). Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

[5] Bezdek, J. C., Ehrlich, R., & Full, W. (1984). Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, *10*, 191–203.

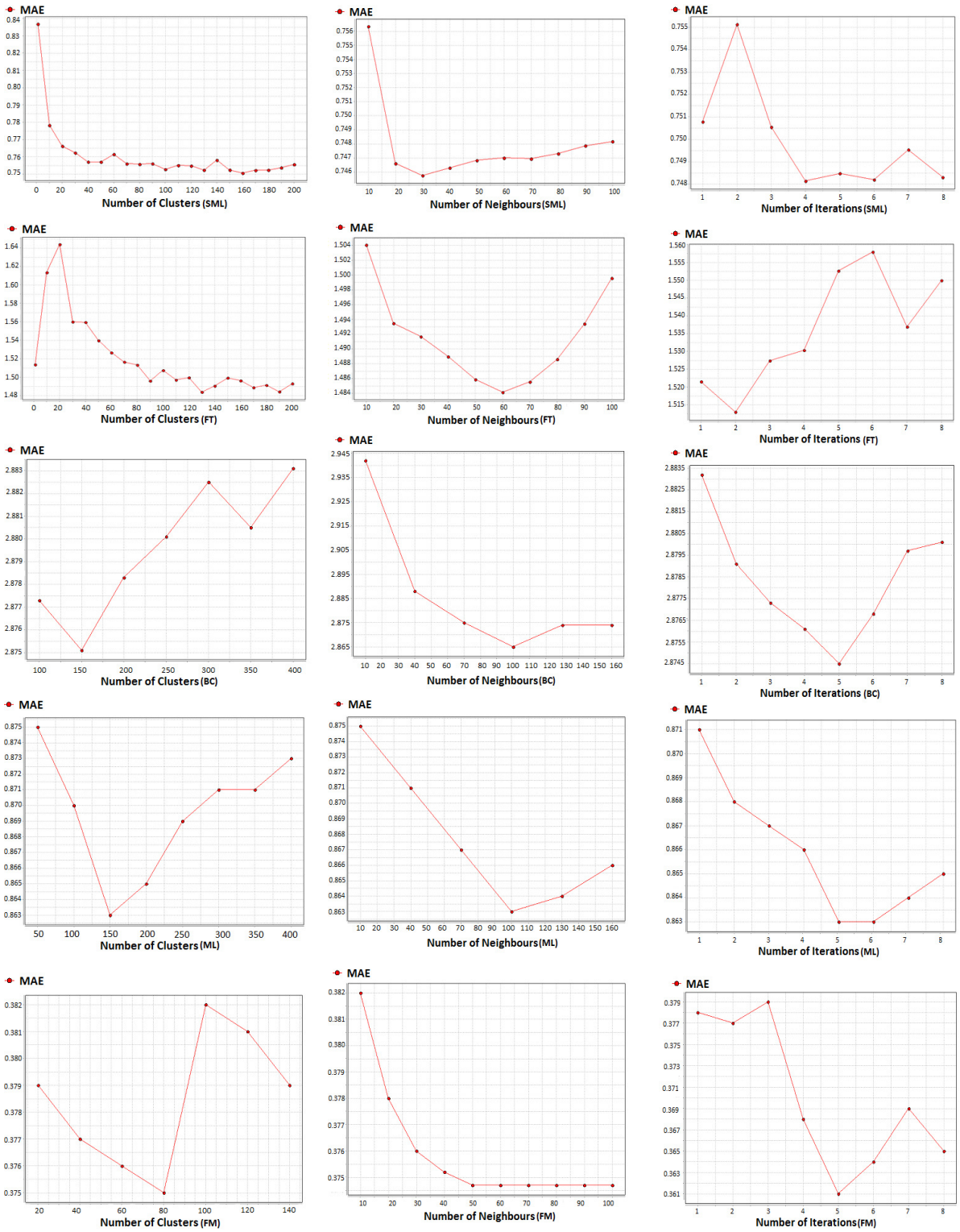
[6] Bhattacharya, A., & De, R. K. (2008). Divisive correlation clustering algorithm (dcca) for grouping of genes: detecting varying patterns in expression profiles. *Bioinformatics*, *24*, 1359–1366.

[7] Bilge, A., & Polat, H. (2013). A scalable privacy-preserving recommendation scheme via bisecting k-means clustering. *Information Processing & Management*, *49*, 912–927.

[8] Bradley, P. S., & Fayyad, U.-a. M. (1998). Refining initial points for K-Means clustering. In *Proc. 15th International Conf. on Machine Learning* (pp. 91–99). Morgan Kaufmann, San Francisco, CA.

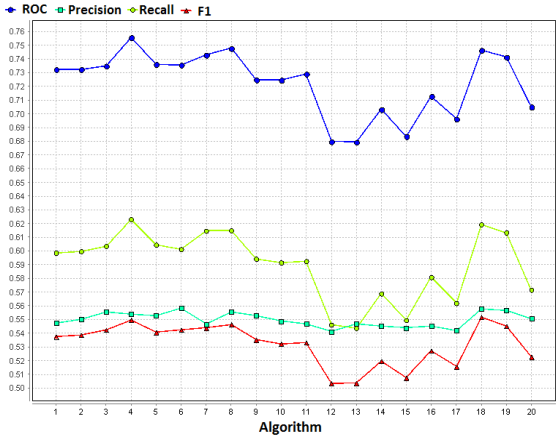
[9] Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. (pp. 43–52). Morgan Kaufmann.

[10] Burke, R. (2007). The adaptive web. chapter Hybrid web recommender systems. (pp. 377–408). Berlin, Heidelberg: Springer-Verlag.

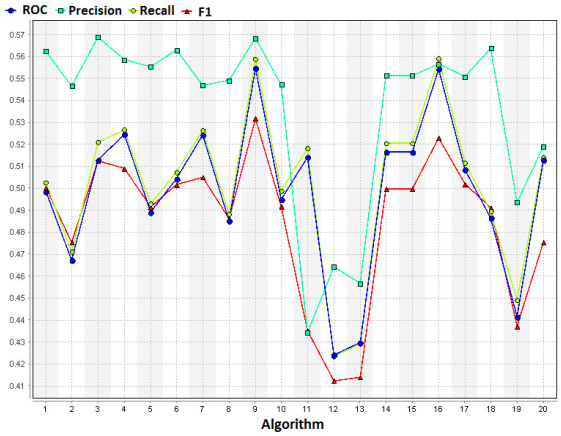


(a) Finding the optimal numbers of clusters (b) Finding the optimal neighbourhood size (c) Finding the optimal numbers of iterations

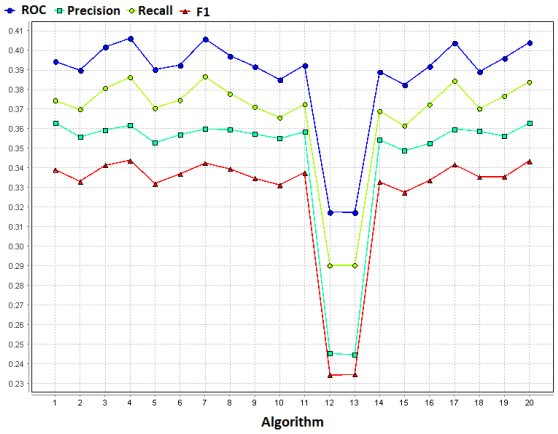
Figure A.8: (From left to right, top to bottom) Determining the optimal number of clusters, neighbourhood size in the Cluster-based CF algorithm (CCF), and number of iterations (itr) in k-means clustering algorithm through the test sets of SML, FT, BC, ML, and FM datasets.



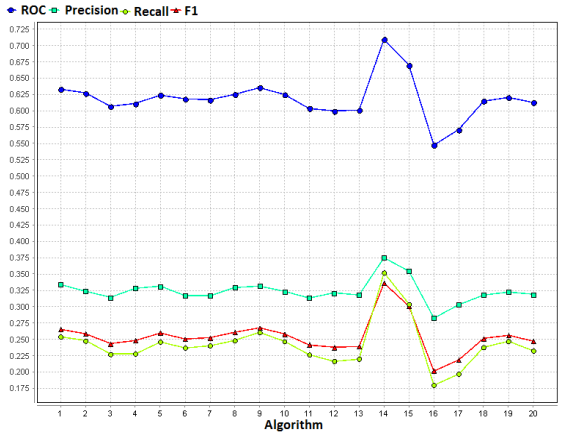
(a) SML Recommendation Results



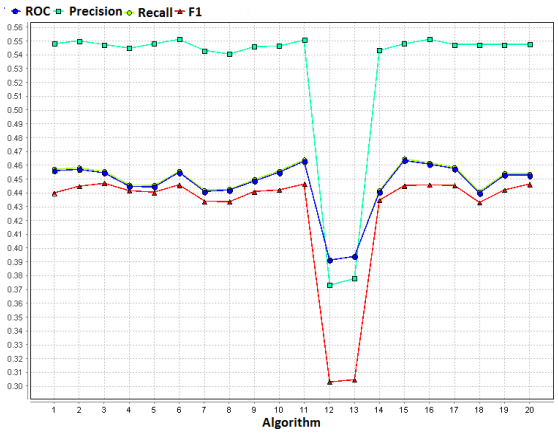
(b) FT Recommendation Results



(c) BC Recommendation Results

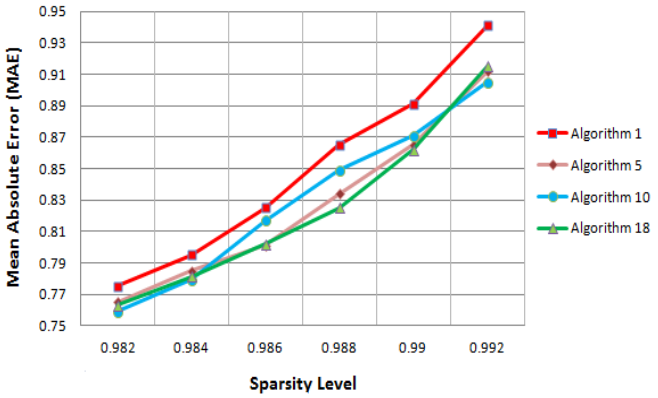


(d) ML Recommendation Results

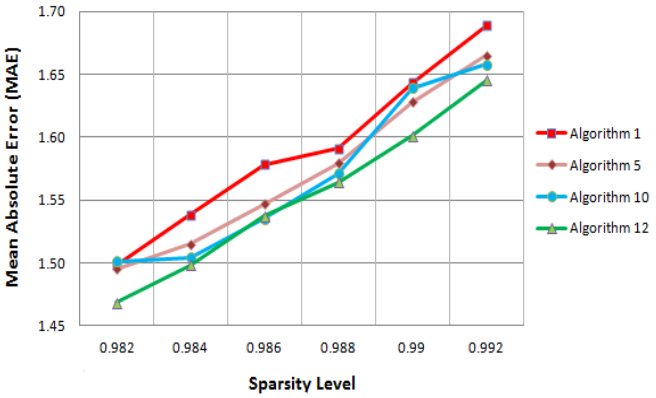


(e) FM Recommendation Results

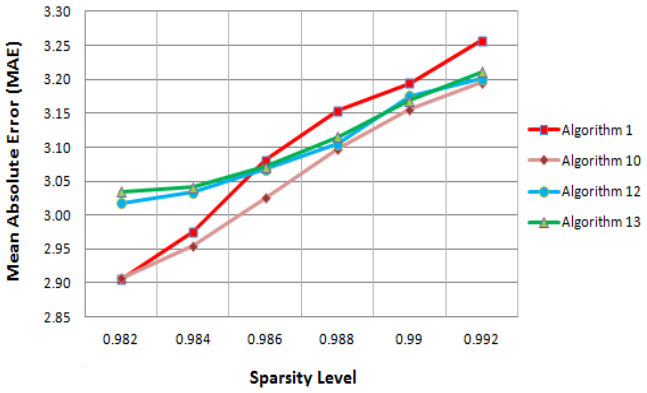
Figure B.9: Inspecting the performance of different centroid selection approaches by measuring ROC-Sensitivity, precision, recall, and F1 over SML, FT, BC, ML and FM datasets. For both SML and BC datasets, results of Algorithm 4 ($KmeansDensity$) and Algorithm 18 ($KMeansUniform$) are comparable, and also better than others, while for FT dataset, results show that Algorithm 9 ($KMeansPlus^{Density}$) outperforms all the rest for all four measures. For ML dataset Algorithm 14 ($KMeansNormal^{Users}$) give best results while for FM dataset results of Algorithm 11 ($KMeansPlus^{Power}$) and Algorithm 6 ($KMeansVariance^{AvgPairwise}$) are comparable as well as better then all other approaches.



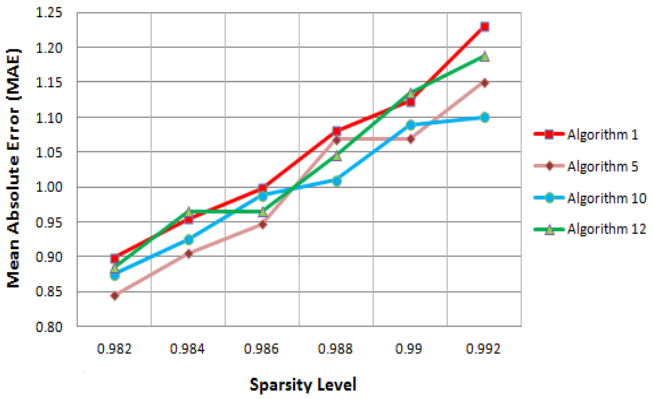
(a) SML Results



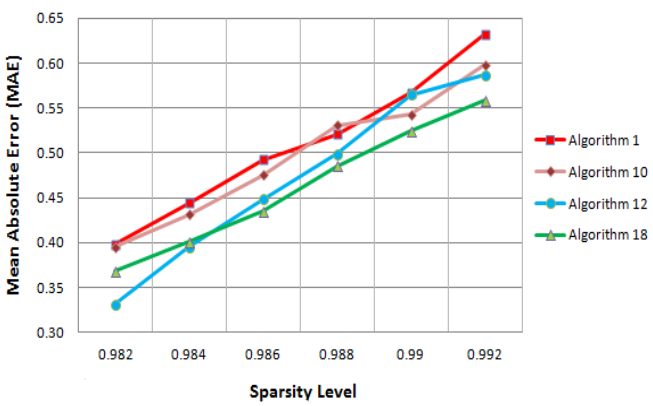
(b) FT Results



(c) BC Results

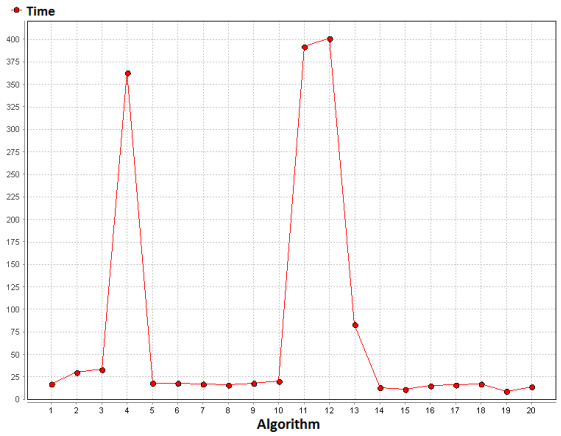


(d) ML Results

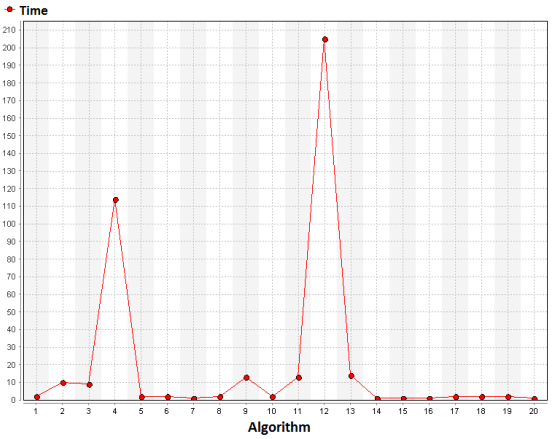


(e) FM Results

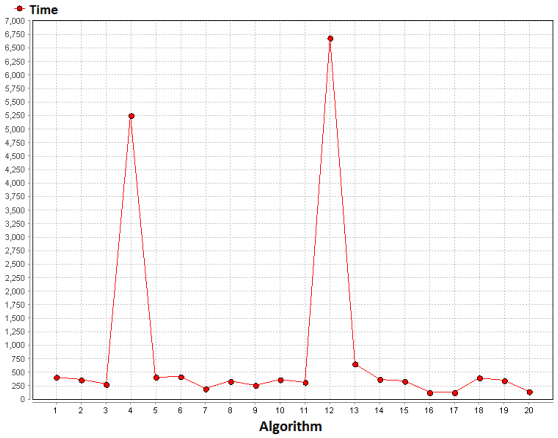
Figure B.10: How the performance of different centroid selection approaches is affected by Sparsity. Results are shown in terms of MAE, which shows that convention k-means (Algorithm 1) suffers the most under sparse conditions, over all five datasets.



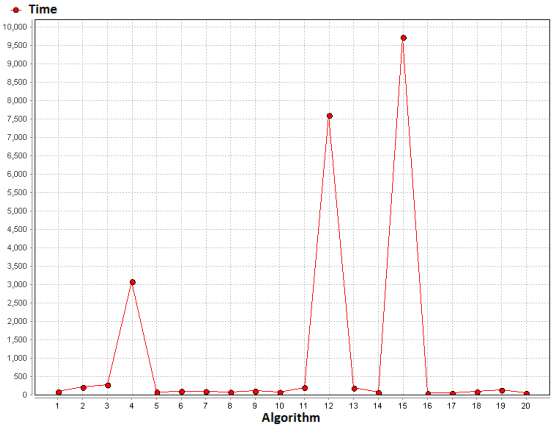
(a) SML Cluster Building Time



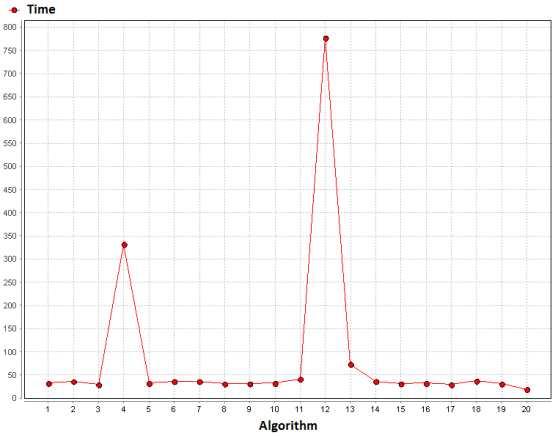
(b) FT Cluster Building Time



(c) BC Cluster Building Time

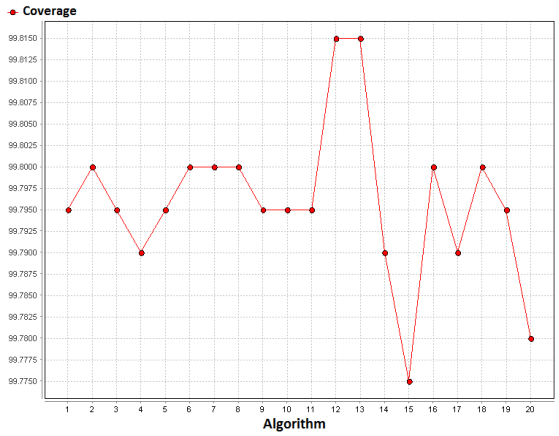


(d) ML Cluster Building Time

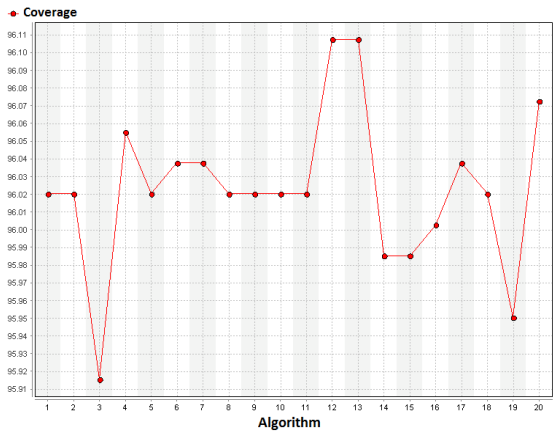


(e) FM Cluster Building Time

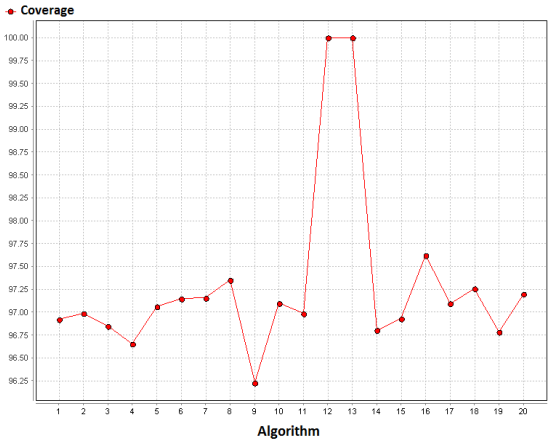
Figure B.11: Comparing the cluster building time of different centroid selection approaches over SML, FT, BC, ML and FM datasets. Different algorithms based on data distributions take minimum cluster building time for all five datasets.



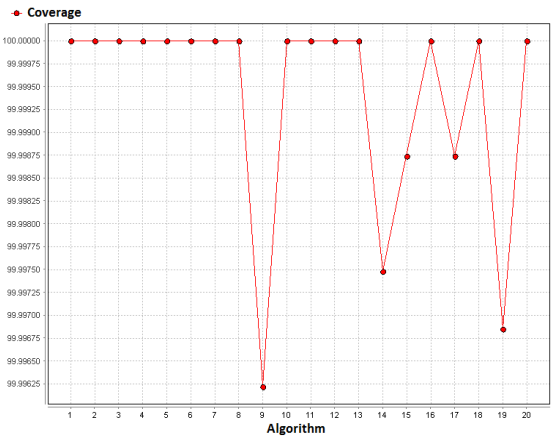
(a) SML Recommendation Coverage



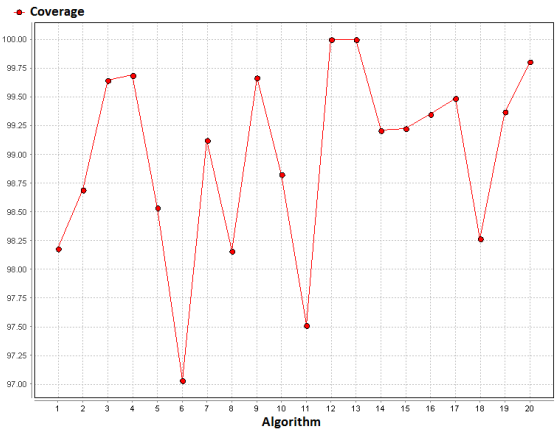
(b) FT Recommendation Coverage



(c) BC Recommendation Coverage



(d) ML Recommendation Coverage



(e) FM Recommendation Coverage

Figure B.12: Comparing the recommendation coverage of different centroid selection approaches, Figure shows that Algorithm 12 ($KMeansPlus^{ProbPower}$) and Algorithm 13 ($KMeansPlus^{LogPower}$) give the maximum coverage for all five datasets.

[11] Cabria, I., & Gondra, I. (2012). A mean shift-based initialization method for k-means. In *CIT* (pp. 579–586). IEEE Computer Society.

[12] Cantador, I., Brusilovsky, P., & Kuflik, T. (2011). 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems* RecSys 2011. New York, NY, USA: ACM.

[13] Claypool, M., Gokhale, A., Mir, T., Murnikov, P., Netes, D., & Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *In Proceedings of ACM SIGIR Workshop on Recommender Systems*.

[14] Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, *39*, 1–38.

[15] Dunn, J. C. (1973). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics*, *3*, 32–57.

[16] Fahim, A., Salem, A., Torkey, F., & Ramadan, M. (2006). An efficient enhanced k-means clustering algorithm. *Journal of Zhejiang University SCIENCE A*, *7*, 1626–1633.

[17] Gedikli, F. (2013). *Recommender Systems and the Social Web: Leveraging Tagging Data for Recommender Systems*. Springer.

[18] Ghazanfar, M. A. (2015). Experimenting switching hybrid recommender systems. *Informatica*, .

[19] Ghazanfar, M. A., & Prügel-Bennett, A. (2010). An Improved Switching Hybrid Recommender System Using Naive Bayes Classifier and Collaborative Filtering. In *Lecture Notes in Engineering and Computer Science: Proceedings of The International Multi Conference of Engineers and Computer Scientists 2010* (pp. 493–502). IMECS 2010, 17–19 March, 2010, Hong Kong.

[20] Ghazanfar, M. A., & Prügel-Bennett, A. (2010). Building Switching Hybrid Recommender System Using Machine Learning Classifiers and Collaborative Filtering. *IAENG International Journal of Computer Science*, *37*, 272–287.

[21] Ghazanfar, M. A., & Prügel-Bennett, A. (2010). A scalable, accurate hybrid recommender system. In *Proceedings of the 2010 Third International Conference on Knowledge Discovery and Data Mining* WKDD '10 (pp. 94–98). Washington, DC, USA: IEEE Computer Society.

[22] Ghazanfar, M. A., & Prügel-Bennett, A. (2013). The advantage of careful imputation sources in sparse data-environment of recommender systems: Generating improved svd-based recommendations. *Informatica*, *13*, 61–92.

[23] Ghazanfar, M. A., & Prügel-Bennett, A. (2013). Exploiting context in kernel-mapping recommender system algorithms. In *Sixth International Conference on Machine Vision (ICMV 13)* (pp. 906727–906727). International Society for Optics and Photonics.

[24] Ghazanfar, M. A., & Prügel-Bennett, A. (2014). Leveraging clustering approaches to solve the gray-sheep users problem in recommender systems. *Expert Systems with Applications*, *41*, 3261–3275.

[25] Ghazanfar, M. A., Prügel-Bennett, A., & Szedmak, S. (2012). Kernel-mapping recommender system algorithms. *Information Sciences*, *208*, 81–104.

[26] Ghazanfar, M. A., Szedmák, S., & Prügel-Bennett, A. (2011). Incremental kernel mapping algorithms for scalable recommender systems. In *ICTAI* (pp. 1077–1084).

[27] Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, *22*, 5–53.

[28] Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Min. Knowl. Discov.*, *2*, 283–304.

[29] Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recogn. Lett.*, *31*, 651–666.

[30] Kaufman, L., & Rousseeuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. (9th ed.). Wiley-Interscience.

[31] Khan, S. S., & Ahmad, A. (2004). Cluster center initialization algorithm for k-means clustering. *Pattern Recogn. Lett.*, *25*, 1293–1302.

[32] Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). Grouplens: applying collaborative filtering to usenet news. *Commun. ACM*, *40*, 77–87.

[33] Kurada, R. R., Pavan, K. K., & Rao, A. V. D. (2013). A preliminary survey on optimized multiobjective metaheuristic methods for data clustering using evolutionary approaches. *CoRR*, *abs/1312.2366*.

[34] Li, Q., & Kim, B. M. (2003). Clustering Approach for Hybrid Recommender System. In *Web Intelligence* (pp. 33–38).

[35] Love, D. (2011). 11 essential algorithms that make the internet work.

[36] Lucas, J. P., Luz, N., Moreno, M. N., Anacleto, R., Figueiredo, A. A., & Martins, C. (2013). A hybrid recommendation approach for a tourism system. *Expert Systems with Applications*, *40*, 3532–3550.

[37] Maitra, R., Peterson, A. D., & Ghosh, A. P. (2010). A systematic evaluation of different methods for initializing the k-means clustering algorithm. *IEEE Transactions on Knowledge and Data Engineering*, (pp. 522–537).

[38] Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. In *in Eighteenth National Conference on Artificial Intelligence* (pp. 187–192).

[39] Melville, P., & Sindhwani, V. (2010). Recommender systems. In *Encyclopedia of Machine Learning* (pp. 829–838).

[40] Milligan, G. W., & Isaac, P. D. (1980). The validation of four ultrametric clustering algorithms. *Pattern Recognition*, *12*, 41 – 50.

[41] Mirkin, B. (2005). *Clustering For Data Mining: A Data Recovery Approach (Chapman & Hall/Crc Computer Science)*. Chapman & Hall/CRC.

[42] Mooney, R., Bennett, P., & Roy, L. (1998). Book recommending using text categorization with extracted information. In *Recommender Systems. Papers from 1998 Workshop. Technical Report WS-98*. volume 8.

[43] Mooney, R. J., & Roy, L. (2000). Content-based book recommending using learning for text categorization. In *Proceedings of DL-00, 5th ACM Conference on Digital Libraries* (pp. 195–204). San Antonio, US: ACM Press, New York, US.

[44] Nazeer, K. A., & Sebastian, M. P. (2010). Clustering biological data using enhanced k-means algorithm. In *Electronic Engineering and Computing Technology* chapter 37. (pp. 433–442). Springer.

[45] Nazeer, K. A. A., Kumar, S. D. M., & Sebastian, M. P. (2011). Enhancing the k-means clustering algorithm by using a o(n logn) heuristic method for finding better initial centroids. In *Proceedings of the 2011 Second International Conference on Emerging Applications of Information Technology* EAIT '11 (pp. 261–264). Washington, DC, USA: IEEE Computer Society.

[46] Park, Y. J., & Tuzhilin, A. (2008). The Long Tail of Recommender Systems and How to Leverage It. In *Proceedings of the 2008 ACM Conference on Recommender Systems* RecSys '08 (pp. 11–18). New York, NY, USA: ACM.

[47] Pena, J., Lozano, J., & Larranaga, P. (1999). An empirical comparison of four initialization methods for the k-means algorithm.

[48] Porcel, C., Tejeda-Lorente, A., Martnez, M. A., & Herrera-Viedma, E. (2012). A hybrid recommender system for the selective dissemination of research resources in a technology transfer office. *Inf. Sci.*, *184*, 1–19.

[49] Rashid, A. M., Lam, S. K., Karypis, G., & Riedl, J. (2006). Clustknn: a highly scalable hybrid model-& memory-based cf algorithm. In *Proc. of WebKDD 2006: KDD Workshop on Web Mining and Web Usage Analysis, in conjunction with the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006), August 20-23 2006, Philadelphia, PA*. Citeseer.

[50] Robinson, F., Apon, A., Brewer, D., Dowdy, L., Hoffman, D.,

& Lu, B. (2006). Initial starting point analysis for k-means clustering: a case study. *Proceedings of ALAR 2006 Conference on Applied Research in Information Technology*, .

[51] Sarwar, B., Karypis, G., Konstan, J., & Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web* (pp. 285–295). ACM New York, NY, USA.

[52] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. (pp. 158–167). ACM Press.

[53] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Application of dimensionality reduction in recommender system–a case study. In *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*. Citeseer.

[54] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2002). Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the Fifth International Conference on Computer and Information Technology*.

[55] Serrano-Guerrero, J., Herrera-Viedma, E., Olivas, J. A., Cerezo, A., & Romero, F. P. (2011). A google wave-based fuzzy recommender system to disseminate information in university digital libraries 2.0. *Inf. Sci.*, *181*, 1503–1516.

[56] Shindler., M. (2008). Approximation algorithms for the metric k-median problem.

[57] Soumi Ghosh, S. K. D. (2013). Comparative Analysis of K-Means and Fuzzy C-Means Algorithms. *International Journal of Advanced Computer Science and Applications(IJACSA)*, *4*.

[58] Tejeda-Lorente, A., Porcel, C., Peis, E., Sanz, R., & Herrera-Viedma, E. (2014). A quality based recommender system to disseminate information in a university digital library. *Inf. Sci.*, *261*, 52–69.

[59] Tzortzis, G., & Likas, A. (2014). The minmax k-means clustering algorithm. *Pattern Recognition*, *47*, 2505 – 2516.

[60] Wen, Q., & Celebi, M. E. (2011). Hard versus fuzzy c-means clustering for color quantization. *EURASIP J. Adv. Sig. Proc.*, *2011*, 118.

[61] Wu, J., & Li, T. (2008). A modified fuzzy c-means algorithm for collaborative filtering. In *Proceedings of the 2Nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition* NETFLIX '08 (pp. 2:1–2:4). New York, NY, USA: ACM.

[62] Wu, K.-L., & Yang, M.-S. (2002). Alternative c-means clustering algorithms. *Pattern Recognition*, *35*, 2267 – 2278.

[63] Xu, R., Wang, S., Zheng, X., & Chen, Y. (2014). Distributed collaborative filtering with singular ratings for large scale recommendation. *Journal of Systems and Software*, *95*, 231 – 241.

[64] Xue, G.-R., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yu, Y., & Chen, Z. (2005). Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* SIGIR '05 (pp. 114–121). New York, NY, USA: ACM.

[65] Yuan, F., Meng, Z.-H., Zhang, H.-X., & Dong, C.-R. (2004). A new algorithm to get the initial centroids. In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on* (pp. 1191–1193 vol.2). volume 2.

[66] Zanardi, V. (2011). *Addressing the cold start problem in tag-based recommender systems*. Ph.D. thesis UCL (University College London).

[67] Zanardi, V., & Capra, L. (2011). A scalable tag-based recommender system for new users of the social web. In *Proceedings of the 22nd international conference on Database and expert systems applications - Volume Part I* DEXA'11 (pp. 542–557). Berlin, Heidelberg: Springer-Verlag.

[68] Zhang, C., & Fang, Z. (2013). An improved k-means clustering algorithm. *Journal of Information Computational Science*, *1*, 193199.

[69] Ziegler, C.-N., McNee, S. M., Konstan, J. A., & Lausen, G. (2005). Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web* WWW '05 (pp. 22–32). New York, NY, USA: ACM.