Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IMPACT FACTOR: 6.017

IJCSMC, Vol. 6, Issue. 6, June 2017, pg.143 – 148

Recommendations Using Modified K-Means Clustering and Voting Theory

Poonam Rani[1], Jyoti Shokeen[2], Dhruv Mullick[3]

[1,3]Department of Computer Science and Engineering, NSIT Dwarka, India

[2]University Institute of Engineering and Technology, MDU Rohtak, India

[1] Poonam.rani.nsit@gmail.com; [2] jyotishokeen12@gmail.com; [3] dhruvm1.co@nsit.net.in

Abstract— Recommender Systems are popularly used for generating user-specific recommendations for services, products and information. In particular, for the process of generating recommendations, a combination of K-Means and Collaborative Filtering algorithm is used extensively. K-Means is a standard data clustering algorithm which suffers from the problem of low recommendation quality due to random cluster center initialization. After generating the clusters, Collaborative filtering is a commonly used technique for generating recommendations. But this technique suffers from the problems of data sparsity and scalability. In this paper, we have used a modified K-Means clustering algorithm for partitioning users into independent clusters. Voting theory has been used to find out the most suited cluster for which the new user can be a part of, and then recommend items to the user. The proposed method attempts to reduce the cost for generating clusters, and to make the clustering method scalable.

Keywords— Recommender Systems; Collaborative Filtering; K-Means; Voting theory.

I. INTRODUCTION

With the huge amount of content available on the internet, it has become necessary to develop a system that recommends an item to the user who is interested in it. When there is a need to make decisions from the vast available options but sufficient personal experience is not available, then we rely on the recommendations based on trusted sources. This is considered as an inevitable outcome of the human decision-making process (1).

As data is growing at an alarming rate it becomes impossible to analyze it simply using human capabilities. Recommender systems play a vital role in such scenario by suggesting the items to the users that interest him.

Basically, recommender systems are based on three approaches: Demographic based filtering (DF) (2), Content based Filtering (CB) , Collaborative Filtering (CF) (3,4). Recently, Hybrid filtering (5) methods have also developed. Content based filtering (CB) produces recommendations for a user on the basis of similarity of the items with which he has interacted with earlier. Demographic filtering (DF) generates the recommendations on the basis of the demographic data of the user, like age, gender and occupation. Collaborative filtering (CF) on the other hand, generates recommendations by taking into consideration the history of the user's ratings of items, and that of the users who are similar to the user under consideration. CF is the technique used in the recommendation systems employed by Amazon and iTunes. Hybrid filtering techniques combine one or more of these filtering techniques to generate recommendations. However, recommendation systems based on CF suffer from a variety of problems like cold start (6), data sparsity and scalability. Cold start is a problem which occurs due to unavailability of data, when a new item or a new user is introduced. Collaborative filtering relies on the history of ratings of users on items to generate recommendations. When a new user is introduced, we may have a "New User cold start problem" (7,8), as we don't have history of this new user's recommendations, and are unable to recommend robustly. When a new item is introduced, the system is not able to ascertain to which user this new item should be recommended. This happens due to the non-rating of this item by any user earlier. This problem is termed as "New Item cold start problem" (9). Scalability is the ability of the recommendation system to scale gracefully as data increases. If a dataset is sparse in nature, it is a problem for a Collaborative filtering system, as there are fewer items rated by every user, and fewer users to whom every item has been recommended. Due to these problems, it becomes challenging to generate useful recommendations.

Clustering is an unsupervised technique of grouping objects into clusters such that objects in one cluster are more similar tan objects of other clusters. A vast research has been done in literature in the field of clustering for the efficient community detection in social networks (10–15). A number of approaches can be used to cluster the objects. So, clustering is a general approach which can be achieved by different techniques. Clustering algorithms are popularly categorized as - Hard clustering algorithms and Soft clustering algorithms (16). Hard clustering algorithms, such as K-Means, partition the data points into hard clusters, i.e. each user is considered as a part of only one cluster. Soft clustering algorithms, like Fuzzy C-Means, partition the users such that each of them can belong to more than one cluster. In such a case, a single user has different membership values for each of the clusters. For example, a user in a social network can belong to more than one community. Such communities where a user belongs to more than one cluster are called overlapped communities (17,18).

In the proposed work, a voting algorithm for the recommender system is proposed. Voting theory has been commonly used for group decision making in multi-agent systems with good results (19). Using voting theory, it is ensured that the system produces recommendations which are in accordance with the preferences of the user. Moreover, it enhances scalability of the system. Further details of the Voting theory can be found in Section II.

II. RELATED WORK

For years, clustering has been a popular problem, for which numerous methods have been developed. Ayed et al. (20) has classified clustering algorithms as partitioning, density-based, hierarchical, dimensionality reduction, graph-based and fuzzy algorithms. Partitioning algorithms include Expectation-Maximisation (21) and K-means (22). Of the clustering methods, most widely used is the K-means algorithm suggested in [13]. In this method, the whole dataset is divided into k disjoint clusters. In this algorithm, we choose *k* random initial points (where *k* is a constant value giving the desired number of clusters to be created) as the initial cluster centroids. In the traditional K-means algorithm (23), the centroid locations are initialized randomly. It has been found that this leads to poor recommendations and a slow convergence speed for clustering, and hence a greater cost. Authors in [2] suggest techniques for centroid selection which can enhance performance as well as save cost. Centroid selection methods exploit data correlation, and show better performance and accuracy, when compared with the random centroid initialization techniques.

Voting algorithms can be used when there are several conflicting alternatives we need to make a choice among them. Voting theory has been used in several applications over the years (24,25). In the work proposed in (24), Voting theory is used in creating a movie recommendation system. The notion of combinatorial vote is introduced by Lang in (26). In this approach, preferences are conveyed by a group of voters, who then come to a mutually acceptable decision regarding the assignment of a few non-independent variables.

III.PROPOSED SCHEME

In the proposed algorithm, we first make use of a modified K-Means algorithm for clustering users. The idea is to divide the users into independent partitions, and apply the recommendation algorithm independently to each of them. Users belonging to the same cluster would have similar preferences. We now use a voting algorithm to take into consideration the preferences of the individual users, and recommend a suitable product. While generating recommendations for a user in a cluster, the system only considers the preferences and ratings of other users in the same cluster. The preferences and ratings of users in other clusters are not relevant and therefore are not taken for consideration.

A known problem with the traditional k-means algorithm is centroid selection. The popular and simple way to handle this problem is to select the initial k centroids randomly. However, this leads to poor recommendation quality, and greater cost for clustering. Authors in (16) suggest the following techniques for centroid selection in K-Means algorithm:

1. KMeansPlus:

Select the first centroid randomly and after that, keep choosing centroids such that they have maximum distance from centroids which are already selected, until the *k* centroids are chosen.

2. KMeansPlusPlus:

Select the first centroid randomly. Then the other *k-1* centroids are chosen on the basis of a probability which is proportional to maximum distance from the already selected centroids.

3. KMeansDensity:

Selects all the initial centroids such that they are well-spaced, and they have a large number of neighbors in a multidimensional spherical representation.

4. KMeansVariance:

Initialise the *k* cluster centroid positions such that they are at differing distances from the mean location.

5. KMeansVariance$^{\text{Avg.\_Pair\_Wise}}$:

It is similar to KMeansVariance, apart from that it defines the mean value to be the average pairwise distance between all the users.

6. KMeansVariance$^{\text{Version}}$:

It is similar to KMeansVariance, but while selecting the centroid positions we measure standard deviation in a different manner.

7. KMeansQuantiles:

Selects the initial cluster centroids as quantiles of the given dataset, which refer to equal increments in probabilities.

8. KMeansPlus$^{\text{Density}}$:

It is a modification of KMeansDensity. Selects the first centroid to be the highest density point. Based on the highest density point, it calculates the minimum distance that separates the centroids. Highest density point is the one closest to the maximum number of points

9. KMeansSortedDistance:

It sorts all data points on the basis of the distances from average rating.

10. KMeansPlus$^{\text{Power}}$:

Applies KMeansPlusPlus algorithm only on the Power users.

11. KMeansPlus$^{\text{Prob\_Power}}$:

Initialises k centroids such that they have probability proportional to the number of ratings given by the users, and the distances.

12. KMeansPlus$^{\text{Log\_Power}}$:

Uses the power users to find k centroids on the basis of the log of similarity and probability proportional to distance with current top power user.

13. KMeansNormal$^{\text{Users}}$:

In this algorithm, a normal distribution is applied over the dataset such that for the mean of the distribution curve, we use the mean rating given by all of the users in the dataset. From the distributed dataset, k centroids are selected arbitrarily.

14. KMeansNormal$^{\text{Movies}}$:

In this algorithm, a normal distribution is applied over the dataset, such that for the mean of the distribution curve, we use the mean rating of all the items. From the distributed dataset, k centroids are selected arbitrarily.

15. KMeansPoisson:

In this algorithm, a Poisson distribution is constructed for the dataset. From the distributed dataset, k centroids are selected arbitrarily.

16. KMeansHyperGeometric:

A hypergeometric distribution is applied over the dataset. From the distributed dataset, k centroids are selected arbitrarily.

17. KMeansUniform:

A uniform distribution is applied over the dataset. From the distributed dataset, k centroids are selected arbitrarily.

18. KMeansUniform$^{\text{Version}}$:

Similar to K-Means Uniform, uniform distribution is applied over the dataset, and arbitrarily k centroids are selected from the distributed dataset.

19. KMeansLog:

For the dataset, a logarithmic distribution is constructed. Then from the distributed dataset, k centroids are selected arbitrarily.

The performances of these algorithms have been compared in (16). We find that algorithm KMeansPlus$^{\text{Log\_Power}}$ proves to be robust algorithm when applied on different datasets. The comparison metrics – Mean Absolute Error (MAE) and Coverage also indicate that this algorithm performs considerably well, and in many cases, the best. MAE gives the average deviation of the recommended value from the true value of an item. In our case, this represents the mean absolute deviation of the predicted ratings from the true ratings. Lower the value of MAE, better the algorithm performs. Coverage gives the number of user-item pairs for which a recommendation algorithm can make prediction. Higher the value of coverage, better the algorithm performs. We propose using the $\text{KMeansPlus}^{\text{Log\_Power}}$ algorithm, as it gives excellent results on the above mentioned metrics and datasets. The algorithm $\text{KMeansPlus}^{\text{LogPower}}$ can be applied as (16):

Input: Users in the training set - U, total number of clusters – k.
Output: The k centroids – $\{d_1, d_2, d_3 \ldots d_k\}$

```
1. Set the number of clusters – k
2. Set some initial value for the centroids.
3. Repeat
4.     Select the next centroid $c_i$ where $c_i$ = user u belonging to U with the probability:
5.         Prob = dist(u) + log( 1+ 1/(p(x))
6. until k centroids are found
7. Return the centroids obtained
```

This algorithm can be applied to the dataset to obtain the cluster centers of the k clusters. The use of above centroid selection methods solve a numerous issues related to scalability, accuracy and performance. The approach also results in faster convergence and better quality clusters. (16) also provides details regarding which algorithm works best for which dataset.

Once we have obtained clusters of users, we use voting theory on each of them (27). However unlike in (27), we shall use a modified K-Means algorithm for clustering the dataset, instead of DBSCAN. DBSCAN algorithm suffers from some disadvantages. The algorithm has much slower convergence speed and doesn't work well when there are clusters of varying densities. These problems are solved in the modified K-means algorithms suggested above, and owing to their various advantages, this proves to be good choice.

Voting theory allows us to further address scalability concerns, and improve the recommendation quality. Moreover, we have better decision making, by selecting the most popular item in the cluster of users. We consider the Borda count voting theory (28), which is a ranked voting method. For a particular cluster, consider the set of voters (users) to be $\{v_1, v_2, v_3, \ldots\}$, and the set of alternatives (items) as $\{a_1, a_2, a_3, \ldots\}$. Every voter has a preference for every alternative, and votes points according to it. Each voter assigns most favored alternative the highest points, and the least favored alternative the least points. The sum of all the votes corresponding to each genre is stored in a vector, corresponding to each cluster.

A. Dataset:

For the performance of experiments, MovieLens (29) dataset is considered. MovieLens is the most used dataset in the evaluation of performance of recommender systems. The dataset guarantees the voting of about 20 items by each user.

B. Testing Methodology

Considering the MovieLens dataset, our system pipeline is as follows:
While recommending movies for a user, we see the preferences of that user.
Let the genres of movies be {Action, Adventure, Horror, Drama}. In the same order, let the user be interested in these genres according to the weights {3, 4, 5, 0}. We normalize these weights for the user to obtain {0.25, 0.33, 0.41, 0}.
2. We select the genre the user is most interested in. From the preference weights vector, we see the new user prefers watching Horror movies.
3. Next we apply clustering to the dataset to partition the users into clusters of users with similar tastes. This is done using a modified K-Means algorithm. Which algorithm to use in particular depends on the use case, and shall vary with the dataset under consideration. Fig.1 shows how the clustered dataset looks after applying modified K means clustering. The genre weight vectors are shown next to the users.
4. On each cluster, we apply voting theory to get a vector of votes corresponding to cluster.

For the two clusters in the dataset: corresponding to cluster A, we have the vote vector as {1.0, 1.2, 1.3, 0.5}; corresponding to cluster B, we have the vote vector as {0.9, 2.2, 0.3, 0.6}. Fig. 2 shows how the genre preference vector corresponding to each cluster, after voting is done.

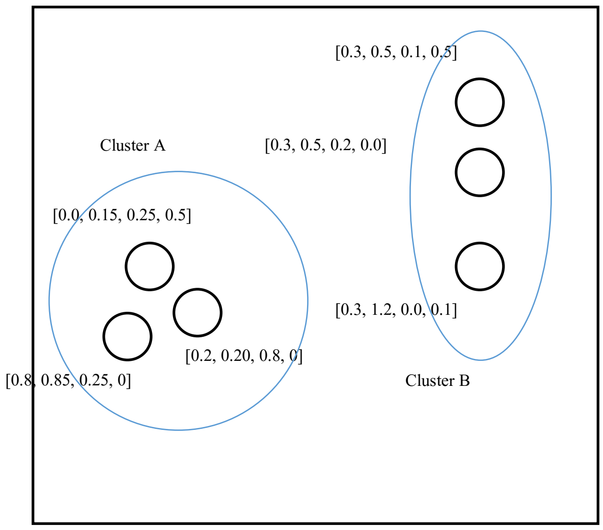

Fig. 1 Clustered dataset, along with genre weights

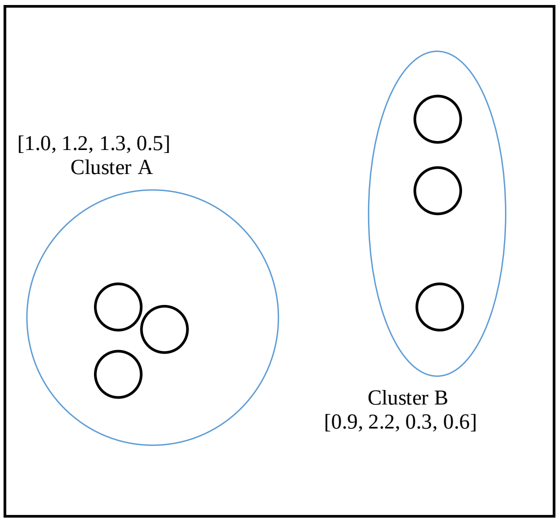

Fig. 2 The view of the Clustered dataset, after voting, along with genre weights corresponding to both clusters

We select the cluster which gave the highest votes to the genre the new user is most interested in. Since our new user prefers the Horror genre the most, we find that he is most suited to be a part of cluster A, which gives the highest rating to the Horror genre. This new user is considered to be a part of this cluster. We recommend the new users movies which corresponding to Horror, on the basis of the recommendations produced from the ratings of the other users in cluster A.

IV. Conclusions and future scope

Though voting theory independently solves the problem of scalability associated with recommendation systems, we still need a good clustering algorithm. A modified K means algorithm serves this purpose, allowing us to cluster the dataset with a low training cost, and excellent recommendation accuracy. Hence, by making use of a hybrid of these two algorithms, we can address scalability concerns, improve accuracy and performance, and get results at a faster rate. Thereby, we have obtained a more robust system for generating recommendations. In the future, we wish to work on other clustering algorithms which can be used with voting theory, as modified K-Means algorithm suffers from the drawback that the number of clusters must be known in advance.

REFERENCES

1. Melville P, Sindhwani V. Recommender Systems. In: Encyclopedia of machine learning. Springer; 2011. p. 829–38.
2. Pazzani MJ. A framework for collaborative, content-based and demographic filtering. Artificial intelligence review. 1999;13:393–408.
3. Ekstrand MD, Riedl JT, Konstan J. Collaborative filtering recommender systems. Foundations and Trends in Human Computer Interaction. 2011;4(2):81–173.
4. Schafer J Ben, Frankowski D, Herlocker J, Sen S. Collaborative filtering recommender systems. The Adaptive Web. 2007;291–324.
5. Burke R. Hybrid Web Recommender Systems. The Adaptive Web. 2007;377–408.
6. Lika B, Kolomvatsos K, Hadjiefthymiades S. Facing the cold start problem in recommender systems. Expert Systems with Applications. 2014;41(4):2065–73.
7. Bobadilla J, Ortega F, Hernando A, Bernal J. A collaborative filtering approach to mitigate the new user cold start problem. Knowledge-Based Systems. 2012;26:225–38.
8. Son LH. Dealing with the new user cold-start problem in recommender systems: A comparative review. Information Systems. 2016;58:87–104.
9. Schein AI, Popescul A, Ungar LH, Pennock DM. Methods and Metrics for Cold-Start Recommendations. In: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. ACM; 2002. p. 253–60.
10. Schaeffer SE. Graph clustering. Computer Science Review. 2007;1(1):27–64.
11. Harenberg S, Bello G, Gjeltema L, Ranshous S, Harlalka J, Seay R, et al. Community detection in large-scale networks : a survey and empirical evaluation. 2014;6(December).
12. Mishra N, Schreiber R, Stanton I, Tarjan RE. Clustering Social Networks. In: International Workshop on Algorithms and Models for the Web-Graph [Internet]. 2007. p. 56--67. Available from: http://www.springerlink.com/index/lrr012003t3757x5.pdf
13. Rana C, Jain SK. An extended evolutionary clustering algorithm for an adaptive recommender system. Social Network Analysis and Mining. 2014;4(1):1–13.
14. Bedi P, Sharma C. Community detection in social networks. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2016;6(3):115–35.
15. Plantie M, Crampes M. Survey on Social Community Detection. Social media retrieval. 2013;65–85.
16. Zahra S, Ghazanfar MA, Khalid A, Azam MA, Naeem U, Prugel-Bennett A. Novel centroid selection approaches for KMeans-clustering based recommender systems. Information Sciences [Internet]. 2015;320(Elsevier):156–89. Available from: http://dx.doi.org/10.1016/j.ins.2015.03.062
17. Amelio A, Pizzuti C. Overlapping Community Discovery Methods: A Survey. In: Social Networks: Analysis and Case Studies [Internet]. 2014. p. 105–25. Available from: http://arxiv.org/abs/1411.3935%5Cnhttp://dx.doi.org/10.1007/978-3-7091-1797-2
18. Wang C, Tang W, Sun B, Fang J, Wang Y. Review on Community Detection Algorithms in Social Networks. In: 2015 IEEE International Conference on Progress in Informatics and Computing (PIC) [Internet]. 2015. p. 551–5. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7489908
19. Ephrati E, Rosenschein JS. Multi-Agent Planning as a Dynamic Search for Social Consensus. IJCAI. 1993;93:423–9.
20. Ayed A Ben, Halima M Ben, Alimi AM. Survey on clustering methods : Towards fuzzy clustering for big data. In: 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR). IEEE; 2014. p. 331–6.
21. Ordonez C, Omiecinski E. FREM : Fast and Robust EM Clustering for Large Data Sets. In: Proceedings of the eleventh international conference on Information and knowledge management. ACM; 2002. p. 590–9.
22. Ghosh S, Dubey SK. Comparative Analysis of K-Means and Fuzzy C- Means Algorithms. International Journal of Advanced Computer Science and Applications. 2013;4(4):35–9.
23. Macqueen J. Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. 1967. p. 281–97.
24. Mukherjee R, Sajja N, Sen S. A movie recommendation system–an application of voting theory in user modeling. User Modeling and User-Adapted Interaction [Internet]. 2003;13(1):5–33. Available from: https://link.springer.com/article/10.1023%2FA%3A1024022819690?LI=true
25. Popescu G. Group Recommender Systems as a Voting Problem. 2013;412–21.
26. Lang J. Logical preference representation and combinatorial vote. Annals of Mathematics and Artificial Intelligence. 2004;42(1):37–71.
27. Das J, Mukherjee P, Majumder S, Gupta P. Clustering-Based Recommender System Using Principles of Voting Theory. In: International Conference on Contemporary Computing and Informatics (IC3I). IEEE; 2014. p. 230–5.
28. Erp M van, Schomaker L. Variants of the Borda count method for combining ranked classifier hypotheses. In: 7th International Workshop on Frontiers in Handwriting Recognition. 2000. p. 443–52.
29. Harper FM, Konstan JA. The movielens datasets: History and context. ACM Transactions on Interactive Intelligent Systems (TIIS) [Internet]. 2016;5(4). Available from: http://dl.acm.org/citation.cfm?doid=2866565.2827872