A Modified Clustering Algorithm DBSCAN Used in a Collaborative Filtering Recommender System for Music Recommendation

Urszula Kużelewska[1] and Krzysztof Wichowski[2]

[1] Bialystok University of Technology,
15-351 Bialystok, Wiejska 45a, Poland
u.kuzelewska@pb.edu.pl
[2] Graduate of the Bialystok University of Technology
vltr@o2.pl

Abstract. Searching in huge amount of information available on the internet is undoubtedly a challenging task. A lot of new web sites are created every day, containing not only text, but other types of resources: e.g. songs, movies or images. As a consequence, a simple search result list from search engines becomes insufficient. Recommender systems are the solution supporting users in finding items, which are interesting for them. These items may be information as well as products, in general. The main distinctive feature of recommender systems is taking into account personal needs and taste of users. Collaborative filtering approach is based on users' interactions with the electronic system. Its main challenge is generating on-line recommendations in reasonable time coping with large size of data. Appropriate tool to support recommender systems in increasing time efficiency are clustering algorithms, which find similarities in off-line mode. Commonly, it involves decreasing of prediction accuracy of final recommendations. This article presents an approach based on clustered data, which prevents the negative consequences, keeping high time efficiency. An input data are clusters of similar items and searching the items for recommendation is limited to the elements from one cluster.

1 Introduction

Recommender systems (RS) are electronic applications with the aim to generate for a user a limited list of items from a large items set. In case of personalised RS the list is constructed basing on the active user's and other users' past behaviour. People interact with recommender systems by visiting web sites, listening to the music, rating the items, doing shopping, reading items' description, selecting links from search results. This behaviour is registered as access log files from web servers, or values in databases: direct ratings for items, the numbers of song plays, content of shopping basket, etc. After each action users can see different, adapted to them, recommendation lists depending on their tastes [6,14,13,2].

Recommender systems are used for many purposes in various areas. The examples are internet news servers (e.g. Google News, http://news.google.com/),

W. Zamojski et al. (eds.), *Theory and Engineering of Complex Systems and Dependability*,
Advances in Intelligent Systems and Computing 365, DOI: 10.1007/978-3-319-19216-1_23

which store the type and content of articles, count the time spent on each of them and update propositions of articles to read after each reading, tourism, where personalisation helps users to plan their journeys, e-shops (e.g. Amazon, http://amazon.com) proposing products, which are the most similar to the content of customers' shopping baskets. They are particularly useful in media services, such as Netflix, LastFM or Spotify, recommending movies, songs, artists or propositions to playlists.

Collaborative filtering (CF) techniques are one of approaches to personalisation, which searches similarities among users or items [1,3]; however only archives of registered users behaviour are analysed. As an example, similar users have mostly the same products in their baskets and similar items are bought by the same customers. They can be classified into model-based and memory-based methods. The first approach builds a model on the ratings, which is then used for generating recommendations. The other approach calculates recommendations by searching neighbourhood of similar users or items in the whole archived data.

Recommender systems face many challenges and problems [2]. They particularly concern collaborative filtering, which is one of the most effective and precise approach. In case of a new visitor,without any recorded profile, an issue called cold-start problem appears. Another problem occurs when a new item is added to the offer. In case of CF methods, it has not been assigned yet to any user and cannot be recommended to anyone [10]. In arbitrary recommender system application, a number of offered items is large, whereas a user during one session visits a few to tens of them. It results in sparsity of input data and lower reliability in terms of measuring the similarity between customers [17]. Finally, however, vitally important challenge in the field of on-line recommendations is scalability [10,18]. RS deal with large amount of dynamic data, however the time of results generation should be reasonable to apply them in real-time applications. The user, while reading news, expects to see next offer for him/her in seconds, whereas millions of archived news have to be analysed.

This paper contains results of experiments on collaborative filtering recommender system, which is based on similarities among items identified a priori as clusters. The set of clusters was generated by modified DBSCAN algorithm with different values of their input parameters and evaluated with respect to their genre homogeneity. Finally, quality of prediction (RMSE) and time efficiency of examined system was calculated and compared to other recommenders: memory-based CF, a recommender system based on k-means clusters [8], SVD model-based approach [19], SlopeOne [9].

2 Description of the Clustering Algorithm Used in the System

Application of clustering algorithms can solve several problems in the field of recommender systems. Clustering is a domain of data mining which had been applied in a wide range of problems, among others, in pattern recognition, image processing, statistical data analysis and knowledge discovery [7]. The aim of

cluster analysis is organising a collection of patterns (usually represented as a vector of measurements, or a point in a multi-dimensional space) into clusters based on their similarity [5]. The points within one cluster are more similar to one another than to any other points from the remaining clusters.

In hybrid recommender systems clusters can be used to increase neighbour searching efficiency. In contrast, in memory-based collaborative filtering to identify neighbours is used kNN algorithm, which requires calculating distances between an active user and the registered all ones. Clusters are generated in off-line phase, which additionally reduces time of neighbours searching. Possessing additional information about users, e.g. demographics attribute values, one can create clusters and solve a new user cold-start problem, by recommendation of items, which are popular in the most similar group.

One of the first approaches, where clustering was used to partition users' preferences is described in [16]. The authors used clusters identified in off-line mode by k-means instead of on-line neighbourhood calculated by kNN method. As similarity measure they used Pearson correlation. Finally, quality of predictions was slightly reduced, however time efficiency and scalability increased significantly.

DBSCAN is one of recent clustering algorithms [4]. It identifies clusters as highly dense groups of points. It is particularly effective in case of arbitrary shaped clusters. For recommendations was used in [12] as initial partitioning on demographic attributes of users.

Another example is a modified method proposed in [15]. Modification concerned similarity computation in a clustering procedure. The authors assumed, that points are similar if they have the same neighbours (see Equation 1).

$$neighbour(x_i) = \{x_j : sim_R(x_i, x_j) \leq Eps, x_i \in I, x_j \in I\} \tag{1}$$

It is assumed the following notation:

- I - is a set of items,
- n_I - is a size of set I,
- x_i, x_j - are items in input data,
- Eps - is a parameter of modified DBSCAN (MDBSCAN) related to minimal similarity threshold,
- U - is a set of users,
- n_U - is a size of set U,
- r - a possible value from a ratings set of range $[0, \ldots, r_{max}]$,
- $U_r(x_i)$ - is a set of the users who rated item x_i at rating r.

Basing on the neighbourhood definition, it can be determined formula of neighbour vectors $\overrightarrow{nb_k}$:

$$\overrightarrow{nb_k}(x_i) = [v_1, \ldots, v_{n_I}]^T \quad \text{, where } v_k = \begin{cases} 1, & \text{if } x_k \in neighbour(x_i) \\ 0, & \text{in the other cases} \end{cases} \tag{2}$$

Similarity between neighbour vectors is calculated by cosine similarity function (Equation 3), whereas between points - using similarity definition from information retrieval (Equation 4).

$$sim_N(x_i, x_j) = \frac{\overrightarrow{x_i} \cdot \overrightarrow{x_j}}{|\overrightarrow{x_i}| \cdot |\overrightarrow{x_j}|} \tag{3}$$

Cosine similarity prefers items, which have more common neighbours thereby reduces influence of noise. Similarity sim_R has higher values for pair of items, which are composed of more the same ratings from users.

$$sim_R(x_i, x_j) = \frac{|\cup_{r=0}^{r_{max}} U_r(x_i) \cap U_r(x_j)|}{|\cup_{r=0}^{r_{max}} U_r(x_i) \cup U_r(x_j)|} \tag{4}$$

The procedure of neighbourhood vectors forming and similarity calculation is illustrated by the following example for data from Table 1.

Table 1. Example ratings data

	Items					
Users	x_1	x_2	x_3	x_4	x_5	x_6
u_1	1	2	3	1	5	3
u_2	1	2	5	5	5	5
u_3	3	3	4	5	4	5
u_4	5	1	1	5	4	1

For the example data similarity between items are as follows:

$$sim_R(x_1, x_2) = \frac{0+0+1+0+0}{3+2+2+0+1} = 0.125 \quad sim_R(x_1, x_4) = \frac{1+0+0+0+1}{3+0+1+0+4} = 0.25$$
$$sim_R(x_2, x_3) = \frac{1+0+0+0+0}{2+2+2+1+1} = 0.125 \quad sim_R(x_2, x_6) = \frac{1+0+0+0+0}{2+2+2+0+2} = 0.125$$
$$sim_R(x_3, x_4) = \frac{0+0+0+0+1}{2+0+1+1+4} = 0.125 \quad sim_R(x_3, x_5) = \frac{0+0+0+1+1}{1+0+1+3+3} = 0.25$$
$$sim_R(x_3, x_6) = \frac{1+0+1+0+1}{2+0+1+2+3} = 0.375 \quad sim_R(x_4, x_5) = \frac{0+0+0+0+1}{1+0+0+2+5} = 0.125$$
$$sim_R(x_4, x_6) = \frac{0+0+0+0+2}{2+0+1+0+5} = 0.25 \quad sim_R(x_5, x_6) = \frac{0+0+0+0+1}{1+0+1+2+4} = 0.125$$

The remaining values are 0. Let $Eps = 0.2$, thereby neighbour vectors are:

$\overrightarrow{nb}(x_1) = [0\ 0\ 0\ 1\ 0\ 0]^T$ $\overrightarrow{nb}(x_2) = [0\ 0\ 0\ 0\ 0\ 0]^T$
$\overrightarrow{nb}(x_3) = [0\ 0\ 0\ 0\ 1\ 1]^T$ $\overrightarrow{nb}(x_4) = [1\ 0\ 0\ 0\ 0\ 1]^T$
$\overrightarrow{nb}(x_5) = [0\ 0\ 1\ 0\ 0\ 0]^T$ $\overrightarrow{nb}(x_6) = [0\ 0\ 1\ 1\ 0\ 0]^T$

Taking into account cosine similarity between them, clusters on example data are composed of 5 points: $C_1 = \{x_3, x_4\}$ (the common neighbour is x_6) and $C_2 = \{x_1, x_5, x_6\}$ (the common neighbour is x_4).

3 Experiments

The recommender system used in the described experiments is composed of the following steps, which technical aspects are presented in Figure 1:

1. Filtering and preprocessing of input data.
2. Generating a set of clustering schemes of vectors of items' values.
3. Evaluation of the clustering schemes and selection the most appropriate partitioning.
4. Generating recommendation for active users.

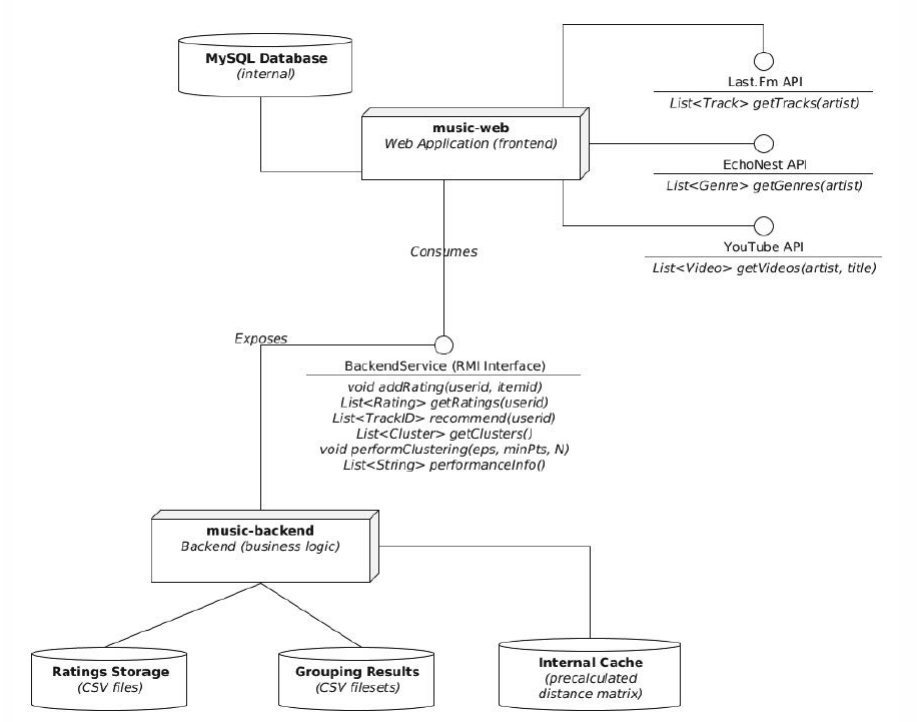

Fig. 1. Components of the recommender system used in the experiments

The main part of the system is *music − backend*, which is an item-based collaborative filtering method implemented in Apache Mahout framework (http://mahout.apache.org). It uses ratings file, clusters and similarity values stored in cache files. A part of application exposed to users is denoted as *music − web*. It gather information from users and external music services (e.g. EchoNest, YouTube, LastFM) and presents them recommendations. Additional database (relational MySQL) stores artists names, tracks titles and users' functional data.

Initial input data for the system were taken from LastFM service (repository at http://dtic.upf.edu.pl/∼oelma/MusicRecommendationDataset/lastfm-1K.html) and contain users' number of song listenings. The higher value of the listening correlates with the higher level of interesting of a particular song. After filtering outdated songs, the data contained 992 users and 2 213 450 songs Then the data were supplemented by users who used the system. Users could listen the music as well as rate it. The ratings were integral values from interval $[1 \ldots 5]$ and expressed how much a user liked the particular song. To unify the numbers of listening with the ratings the following procedure was performed. If a user listened a song only once, it is assumed, that he/she didn't like it. The related rating is 1. The following values were correlated with the numbers of listening with assumption of even decomposition of the numbers in ratings (see Table 2).

Table 2. Decomposition of listening numbers in ratings

Rating value	Range of listening numbers	Numbers of listening
1	1	1 850 078
2	2	667 612
3	3-4	588 676
4	5-8	651 398
5	> 9	642 311

The first evaluation concerned homogeneity of clusters with respect to kind of music of the songs belonging to them. Table 3 contains the best results satisfying these requirements selected from data. High homogeneity is not strictly correlated with the parameters of the MDBSCAN, however there were some values sets connected with the best results: high homogeneity and appropriate number of clusters. Very small number of groups are not desirable, because the neighbourhood search space is not very limited in this case. On the other hand, it is not possible to precisely estimate preferences basing on numerous tiny clusters.

Finally, parameter *Eps* was limited to range [0.3,0.42]. Results on values less than 0.3 were composed of very small (the most often size was equal 1) numerous clusters. Values greater than 0.42 lead to a few large clusters. The most reasonable value of $MinPts$ was 5-8 for the reasons mentioned above. Size of neighbourhood N had to be quite large: greater than 7000 objects.

Another important issue was the ratio of input data which were located within clusters. The highest values was 20.43%, which was not very satisfying.

To increase the number of clustered input data a procedure of complement clustering was performed [15]. For every not clustered point its distance to the formed previously clusters was calculated according to Equation 5.

$$sim_N(x_i, C_j) = \max(sim_N(x_i, x_k)), \text{, where } x_k \in C_j \tag{5}$$

Points are joined to the nearest group, if its distance is closer than parameter γ. This parameter has to be much smaller than *Eps* and its appropriate value is

Table 3. Clustering results of songs from LastFM dataset

Eps	MinPts	N	A number of groups	Ratio clustered data [%]	Homogeneity [%]
0.3	7	8300	1911	5.75	19.14
0.42	7	8300	11912	20.43	77.67
0.39	7	8300	2484	8.18	97.20
0.39	5	8200	6533	9	97.20
0.41	7	8250	5115	13.39	97.47
0.41	7	8300	89	0.47	97.48
0.4	7	8300	5122	13.3	98.96
0.35	8	7100	1209	1.43	98.97
0.35	5	8200	6573	8.91	98.97
0.34	6	7200	1255	1.49	99.09

selected during experiments. Table 4 contains results of complementary clustering with $\gamma = 0.05$. The best result contains 91.47% of input data in clusters for the parameters set: $Eps = 0.8$, $MinPts = 10$ and $N = 8300$ with homogeneity greater than 99%. Number of clusters in this case was equal 48619.

Table 4. Clustering results of items LastFM dataset with complement clustering

Eps	MinPts	N	A number of groups	Ratio clustered data [%]
0.39	7	8300	11964	20.85
0.4	7	8300	11954	20.86
0.7	10	8300	8246	46.67
0.75	10	8300	7808	57.99
0.8	10	8300	48619	91.47

The second evaluation of clustering was performed in item-based collaborative filtering recommender system, in which the best results was used as input data. Table 5 contains RMSE values and time of recommendations calculation for MDBSCAN as well as for other recommender systems used for comparison. It concerns item-based collaborative filtering (CFIB), user-based collaborative filtering (CFUB) with k-nn procedure (k=10 and k=1000) for neighbours searching, SlopeOne algorithm, SVD based recommender. This table contains also comparison to another clustering-based collaborative filtering recommender system. The clustering method was k-means with selected number of clusters: ncl=20 and ncl=1000. In all the mentioned methods as a similarity measure cosine-based was selected. The methods from Table 5 marked with * were examined on smaller dataset due to time or memory problems on greater data.

The values of RMSE errors were computed for every objects from input dataset estimating for them 30% of randomly removed existing preferences and

Table 5. RMSE results on LastFM dataset

Method	Parameters	RMSE	Time of recommendation [s]
MDBSCAN	Eps=0.8, MinPts=10, N=8300	0.22	0.042
MDBSCAN	Eps=0.75, MinPts=10, N=8300	0.63	0.031
CFIB		0.58	118.77
CFUB*	k=10	1.22	0.19
CFUB*	k=1000	1.09	1.04
SlopeOne*		0.68	48.87
SVD*		0.58	69.81
CF k-means	ncl=20	0.71	0.019
CF k-means*	ncl=1000	0.64	0.02

comparing the estimated ones to the real ratings. In every system in this experiments a cosine similarity measure was applied. Time of recommendations generation was calculated for every user from input data with length of recommendation lists equal 7.

The fastest recommender systems were methods based on k-means clusterings, however their RMSE values were higher than in case of MDBSCAN for different values of its parameters. The best results (RMSE=0.22) were generated for MDBSCAN and its input parameters: $Eps = 0.8$, $MinPts = 10$ and $N = 8300$. The methods: CFIB and SVD-based generated good results, however the time of recommendations was inappropriately high (more than 60 s).

4 Conclusions

Recommender systems become an important part of internet services effectively supporting people with searching interesting products or information in huge amount of data. Collaborative filtering approach to this problem faces many challenges. One of vital issue is poor scalability.

In this article a clustering approach to CF recommendations is presented as one of solutions to scalability problem. Clusters were generated using modified DBSCAN method and given to input of collaborative item-based recommender system. As a result, in on-line stage of recommendations generation, searching similarities of an active users' favourite songs was limited to the clusters they

belong to. Finally, time effectiveness of the system increased. Additionally, prediction ability of the method also increased in comparison to techniques such as: memory-based collaborative filtering user-based and item-based systems and hybrid recommenders using k-means clusters.

Acknowledgements. This work was supported by Rectors of Bialystok University of Technology Grant No. S/WI/5/13.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering 17(6), 734–749 (2005)
2. Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A.: Recommender systems survey. Knowledge-Based Systems 46, 109–132 (2013)
3. Candillier, L., Meyer, F., Boullé, M.: Comparing state-of-the-art collaborative filtering systems. In: Perner, P. (ed.) MLDM 2007. LNCS (LNAI), vol. 4571, pp. 548–562. Springer, Heidelberg (2007)
4. Ester, M., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, pp. 226–231. AAAI Press (1996)
5. Jain, A.K., Murty, M., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys 31(3), 264–323 (1999)
6. Jannach, D., et al.: Recommender systems: an introduction. Cambridge University Press (2010)
7. Kużelewska, U.: Advantages of Information Granulation in Clustering Algorithms. In: Filipe, J., Fred, A. (eds.) ICAART 2011. CCIS, vol. 271, pp. 131–145. Springer, Heidelberg (2013)
8. Kużelewska, U.: Clustering Algorithms in Hybrid Recommender System on MovieLens Data. Studies in Logic, Grammar and Rhetoric 37(1), 125–139 (2014)
9. Lemire, D., Maclachlan, A.: Slope One Predictors for Online Rating-Based Collaborative Filtering. In: Proceedings of SIAM International Conference on Data Mining, vol. 5, pp. 1–5 (2005)
10. Lika, B., Kolomvatsos, K., Hadjiefthymiades, S.: Facing the cold start problem in recommender systems. Expert Systems with Applications 41(4), 2065–2073 (2014)
11. Luo, X., Xia, Y., Zhu, Q.: Incremental collaborative filtering recommender based on regularized matrix factorization. Knowledge-Based Systems 27, 271–280 (2012)
12. Moghaddam, S.G., Selamat, A.: A scalable collaborative recommender algorithm based on user density-based clustering. In: 3rd International Conference on Data Mining and Intelligent Information Technology Applications, pp. 246–249 (2011)
13. Park, D.H., et al.: A literature review and classification of recommender systems research. Expert Systems with Applications 39(11), 10059–10072 (2012)
14. Ricci, F., et al.: Recommender Systems Handbook. Springer (2010)
15. Rongfei, J., et al.: A new clustering method for collaborative filtering. In: International IEEE Conference on Networking and Information Technology, pp. 488–492 (2010)
16. Sarwar, B., et al.: Recommender Systems for Large-Scale E-Commerce: Scalable Neighborhood Formation Using Clustering. In: 5th International Conference on Computer and Information Technology (2002)

17. Sarwar, B., et al.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th ACM International Conference on World Wide Web, pp. 285–295 (2001)
18. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advances in Artificial Intelligence 4, 1–19 (2009)
19. Zhang, S., et al.: Using singular value decomposition approximation for collaborative filtering. In: Seventh IEEE International Conference on E-Commerce Technology, CEC, pp. 257–264 (2005)