Journal of Statistical Mechanics: Theory and Experiment
An IOP and SISSA journal

PAPER: Interdisciplinary statistical mechanics

Diffusion-based location-aware recommender systems

Hao Liao[1,2,3], **Xiaojie Zhang**[1], **Zhongtian Long**[4], **Alexandre Vidmer**[1], **Mingkai Liu**[1] and **Mingyang Zhou**[1]

[1] National Engineering Laboratory for big data computing systems, Guangdong Province Key Laboratory of Popular High Performance Computers, Guangdong Province Engineering Center of China-made High Performance Data Computing System, Shenzhen City Key Laboratory of Service Computing and Application, College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, People's Republic of China
[2] Institute of Big Data Intelligent Management and Decision, Shenzhen University, Shenzhen 518060, People's Republic of China
[3] The State Key Laboratory for Management and Control of Complex Systems (SKL-MCCS), the Institute of Automation, Chinese Academy of Science, Beijing 100190, People's Republic of China
[4] DUT-RU International School of Information Science & Engineering at DUT, Dalian, 116000, People's Republic of China
E-mail: alexandre@vidmer.com (Alexandre Vidmer) and zmy@szu.edu.cn (Mingyang Zhou)

Received 27 August 2019
Accepted for publication 11 January 2020
Published 8 April 2020

Online at stacks.iop.org/JSTAT/2020/043401
https://doi.org/10.1088/1742-5468/ab74c5

Abstract. The recommendation is now part of our daily life. As the years pass by, companies collect more and more information about the users of their platforms. One question which could arise is: are the data collected useful for better predictions? In this paper, we investigate the performance impact on adding geographical positions on the performance of the prediction of users' behavior using an existing diffusion-based recommender system. We show how we can improve the accuracy of the diffusion algorithm using the geographical position of users. The accuracy of the improved algorithm is compared with the state of art similar recommender algorithms. Moreover, we design a general framework to infer the position location of users based on the position of their activities.

Keywords: recommender system, privacy, fusion framework, social networks

1742-5468/20/043401+19$33.00

Contents

J. Stat. Mech. (2019) 043401

1. Introduction

Past years have seen a huge increase in the availability of data. Not only in the amount of data, but also in their diversity. One of the first popular challenges to the recommender system was the Netflix Prize in 2009[5]. Since then, many companies have hosted their own challenge, asking the external teams to provide the best algorithm to predict from their data. One of the most famous platforms to host such a challenge is Kaggle[6]. The Netflix Prize consisted of users giving the rating to movies. Additionally, the time of the rating was provided. These properties mentioned above have been successfully applied to various recommender systems [1–3]. Now, the challenge moved to process additional data, such as geographic location, or even analyzing the text content of the reviews [4–6].

Nowadays, there are various datasets with geographic location available. In this work, we use the data from Yelp, Foursquare, and Brightkite. Those three online services are all related to geographical location, Yelp and Foursquare aim to recommend

---

[5] www.netflixprize.com/
[6] www.kaggle.com

the commercial location to users, while Brightkite is a service allowing people to check-in at a different location and share their check-ins with other users. These online services usually have a large amount of data. For instance, Yelp collected more than 10 million reviews between 2004 and 2010 and totaling about 30 million visitors per month [7]. Users need to choose their favorite one from thousands of businesses on the Yelp website. Obviously, two important criteria for their choices are the reviews written by other users and the distance between the business and themselves. A person who lives in Hong Kong would probably not be interested in a recommendation of a restaurant located in New York.

In big towns such as New York, there are thousands of restaurants. Even one lifetime would not be sufficient to try out all these different places. Before the advances in data science, we had to rely on people's word of mouth or the opinion of a single journalist. Nowadays, thanks to collaborative platforms we can rely on the community to filter out the good places from the bad ones. However, the community alone is not sufficient [8, 9]. We also need to rely on algorithms to find the most suitable place. Those algorithms are the building blocks of the *recommender systems* [10]. Recommender systems attempt to predict the users' future behavior based on their past choices, such as browsing, collecting, reviewing, consumption and so on [11, 12].

Since the 1990s, location-based services and social networks started to emerge, as well as location-based social networks [13, 14]. Since the emergence of these networks, recommender systems started to use geographic information [15–17]. The recent work [18] embraced a bunch of Markov-based predictors and a series of location recommendation algorithms to mine location-based social networks. Another work related to location-based recommendation combined collaborative filtering with location [19]. The resulting method was a trade-off between recommendation accuracy and computational effectiveness. In [20], a different approach was chosen. The geographical areas were modeled as a two levels systems, local scale and regional scale were both considered. The number of network-based recommendation algorithms using location is still limited [21]. The additional geographical information and social information such as the social relationships among the users and the similarity between users have been studied on their impact on new venues recommendation and exhibited a great improvement over traditional collaborative filtering [22].

In this work, we are interested in the study and improvement of a diffusion algorithm that was developed based on combining the principle of heat conduction and random walk on a network representation of data [23, 24]. The use of these two diffusion techniques represents two different complementary approaches to compute the similarity between nodes. We show a method to improve the recommendation performance by combining the diffusion algorithm using geographical location data.

This paper is organized as follows: first, the methods used in this work are described. The network framework and notations are given, then we define the metrics used in this work to assess the performance of the recommendation algorithms. We follow the description of the recommendation algorithm. After that, we show how we compute the position of users based on their activities. This is crucial as in most applications the users would not give their home address, and also some users might have activities that are far from their home. We then discuss how we combine the users' position with the recommendation algorithm and finally show how it improves the results.

2. Recommendation metrics and datasets

We start by describing the general network framework and the notation system we use in this paper, as well as the metrics used to evaluate the performance of recommender systems. We then focus on the description of the recommender system, followed by the method used to assess the location of users in the system. Lastly, we show how we include the location of users in the recommendation process.

Many recommender systems related studies divide the data randomly by hiding 10% of the data and trying to predict it based on the remaining 90%. Recently, it was shown that it is essential to keep the time ordering of the data and divide data according to their time-stamp [25], as we are otherwise discarding major features of the data evolution, such as user interest evolution.

J. Stat. Mech. (2019) 043401

2.1. Networks and metrics

Most of the data can be represented as networks (or called Graph). Networks are made of nodes, and two nodes are connected by a link when there is a relation between them in [24]. In this work, we use data with two types of nodes: users and items. A link can only occur between a user and an item (i.e. not between two users or two items). This is what we define as a bipartite network. For the sake of simplicity, we use the generic term of the item for non-user nodes, but they can be anything that interacts with a user. For example a review of a business, a purchased product, a message on an online board.

2.1.1. Notation. We use Latin letters for users and Greek letters for items. As the data are time dependent, most of the quantities depend on the considered time t. The number of users in the dataset is defined as U and the number of items as I. The most fundamental quantity of the network representation is the adjacency matrix $\mathbf{A}$. The elements of $\mathbf{A}$ denote the relations between users and items. We have $a_{i\alpha} = 1$ if user i is connected to item α, and $a_{i\alpha} = 0$ otherwise. The degree of users $k_i(t)$ of user i is the number of links connected to it at time t. Similarly, the degree $k_\alpha(t)$ of item α is the number of users connected to item α.

2.1.2. Recommendation metrics. To evaluate a recommender system, the data are usually divided into two separate sets: the training set and the test set. The algorithm is trained on the training set and then evaluated on the test set. Note that during the training, the algorithm has no access to the information contained in the test set: it attempts to predict it at best without any knowledge of its content. As shown in [25], it is better to divide the data based on time rather than random division when the time is available, as it reflects the real evolution of the data [26]. In practice, we proceed as follows. Consider that the data span from time 0 to time T_m and a probe timespan parameter ΔP. We randomly select a time $T_P \in [0.8T_m, 0.9T_m]$. We set every data with a timestamp $t \leqslant T_P$ as being in the training set, and every data with a timestamp $T_P < t \leqslant T_P + \Delta P$ as the test set. The choice of ΔP depends on the typical time duration of the data. To obtain statistically relevant results, 200 different random T_P are chosen for each evaluation.

The goal of the recommendation system is to make prime suggestions to users. Based on the training set, the recommendation algorithm computes a score for each user-item couple in the network. For each user, the items are ranked from highest score to lowest score, and for each user, the top-N items with the highest score constitute its recommendation list.

We use the following metrics to evaluate the performance of the recommender system: Precision and Recall [27, 28], F1 score [29], AUC [30, 31], NDCG [32], mean distance of recommended items, Novelty [24, 33], and Coverage [34]. The first four metrics measure the accuracy of the algorithm, and Novelty and Coverage measure its diversity. A good algorithm should benefit both users and item providers. On one hand, it should be accurate and diverse on the content it recommends while on the other hand, it also should devote to promote the Coverage for items providers so that no provider is left out.

Recall and Precision [27, 28, 35] both measure the proportion of items that are predicted accurately. If an item α is in the recommendation of user i, and user i collects the item α in the test set, we count the item as correctly predicted. On the contrary, if user i collects the item α in the test set and item α is not in the recommendation list of user i, we count the item as not predicted. If we label c_i the number of correct prediction for user i and n_i the number of items that were not predicted, Recall writes as:

$$R = \frac{1}{U}\sum_{i}^{U}\frac{c_i}{c_i + n_i}. \tag{1}$$

Precision is computed as:

$$P = \frac{1}{U}\sum_{i}^{U}\frac{c_i}{N}, \tag{2}$$

with N the length of the recommendation list.

F1 score [29] combines Precision and Recall. The recommender results hope that the higher the Precision, the better the Recall, but in fact the two are contradictory in some cases. For example, if we search only one result and it is accurate, then Precision is 100%, but Recall is very low; and if we return all the items, Recall is 100%, but Precision is very low. Therefore, they need to be considered comprehensively. F1 is the weighted harmonic average of Precision and recall. The calculation formula is as follows:

$$F1 = \frac{2PR}{P + R}. \tag{3}$$

AUC [30, 31] is a commonly used metric to evaluate the accuracy of prediction algorithms. To explain the idea behind this metric, let us focus on a specific user i. For this user, we select two items randomly, one which the user collects in the test set ($a_{i\alpha}(T_P + \Delta P) = 1$) and one he does not ($a_{i\beta}(T_P + \Delta P) = 0$). If our recommendation algorithm is good, it should give a higher score to item α than β. In our case, the number of samples is quite limited, so we can compute AUC in the following way:

$$\mathrm{AUC} = \frac{1}{U}\sum_{i}^{U}\left(\frac{\sum_{\alpha\in\mathcal{P}_i} y_{i\alpha} - C_i(C_i+1)/2}{C_i N_i}\right), \tag{4}$$

where $\mathcal{P}_i$ is the set of collected items in the test set for user i, and $y_{i\alpha}$ is the rank of item α in the recommendation list for user i, the item with the lowest score being rank 1 and the one with the highest score rank being N. C_i is described as the number of collected items for user i in the test set. On the contrary, N_i represents the number of uncollected items for user i.

NDCG [32] is usually used to measure the ranking quality of the recommendation lists. The higher ranking of items chosen by users in the recommendation list, the higher NDCG score, which means the recommendation result is better. Cumulative gain (CG) accumulates the relative score of each recommendation list and can be written for user u as

$$CG_u(N) = \sum_{i}^{N} \mathrm{rel}_u^i, \tag{5}$$

where rel_u^i denotes the relevance of recommended items at position i in user u's recommendation, and N is the length of the recommendation list. rel_u^i can take any value in the general expression of equation (5). In our work, the rel_i represents the presence or not of the item in the user's test list with a binary value: 1 if the user collects the item, and 0 if not. One disadvantage of CG is that it does not consider the rank of recommended items. However, the results with high relevance should be at the top of the recommendation list. Therefore, discounted cumulative gain (DCG) was created to take the rank into account. The discounted version of CG for user u writes

$$\mathrm{DCG}_u(N) = \sum_{i}^{N} \frac{2^{\mathrm{rel}_u^i} - 1}{\log_2(i+1)}. \tag{6}$$

DCG still has its shortcomings. It is difficult to evaluate the quality of the recommendation between different lists. Normalized discounted cumulative gain (NDCG) introduces a normalization that allows a relevant comparison between the recommendation list of all users. Before introducing NDCG, we need to introduce Ideal DCG (IDCG), which refers to the best possible recommendation list a user, that is, assuming that the recommendation list contains all items from the test set at the top. NDCG can be computed for user u as:

$$\mathrm{NDCG}_u(N) = \frac{\mathrm{DCG}_u(N)}{\mathrm{IDCG}_u}. \tag{7}$$

Finally, NDCG is computed as follows:

$$\mathrm{NDCG}(N) = \frac{1}{U}\sum_{u}^{U} \mathrm{NDCG}_u(N). \tag{8}$$

Novelty [24, 33] refers to the ability of recommender systems to recommended items which are not yet popular to users. It is simple to assume that the lower the item degree is in the recommended list, the better the Novelty should be. The self-information

https://doi.org/10.1088/1742-5468/ab74c5

of the item is used to compute the Novelty of the item. If a user chooses an item at random, the probability of item α being selected is $k_\alpha(t)/U$ The calculation of Novelty is as follows:

$$\text{Novelty} = \frac{1}{U}\sum_{i}^{U}\frac{1}{N}\sum_{\alpha\in\mathcal{R}_i}\log_2\left(\frac{k_\alpha(t)}{U}\right)^{-1}, \tag{9}$$

where $\mathcal{R}_i$ is the recommendation list of user i.

Mean distance is the metric to measure the distance between recommended items and users, i.e. the average distance between users and their products in the recommendation list. In many situations, people tend to prefer closer locations. So the smaller the mean distance, which indicates that the objects in the recommended list are closer to the user, the possibility of user's selection is greater, and the recommendation system performance is better. Mean distance is calculated as follows:

$$\text{Mean Distance} = \frac{1}{U}\sum_{i}^{U}\frac{\sum_{\alpha\in\mathcal{R}_i} d_{i\alpha}}{N}, \tag{10}$$

with $d_{i\alpha}$ the distance between user i and item α defined in section 3.2.

Coverage [34] is described as the proportion of items recommended for all users over the total number of available items. This metric reflects the ability of the algorithm to cover its catalog of items. If the Coverage is higher, it means that more distinct items are recommended, while a lower value indicates that the algorithm recommends a smaller part of the whole set of items. The higher the portion of items covered, the more likely users will discover more items. It also shows the fairness to the item providers. Note that more items are discovered by the users which will increase the popularity of overall products. More importantly, it will affect the item information spreading among users. Coverage is formulated as:

$$\text{Coverage} = \frac{|\cup_{i\in U}\mathcal{R}_i|}{I}, \tag{11}$$

with I the total number of different items in the dataset.

2.2. Empirical datasets

We conduct three different real datasets for our experiment shown in the table 1. The description of the three datasets as follow.

Yelp[7] is a famous business commenting website, which covers businesses such as clothing, food, housing and so on. Users can review and rate the business on the website. Yelp company organized the challenge and released the relevant dataset. In this dataset, it records the links of users' comments, including the time of comments and the geographic location of businesses.

Foursquare[8] is a mobile service website based on user location information. Users can share their location information with others. Using foursquare services, users can not only check in the locations which are already registered in the dataset but can also check in a new location. The locations can be for instance hotels or stores. In this work,

---

[7] www.yelp.com/dataset

we use the dataset in Tokyo from 2012 to 2013. Each link contains user, location of the check-in and time information.

In **Brightkite**[9], users can publish text and photos on the site and check in somewhere, and others can comment on them, so they can get to know new friends according to where they go. This paper used the dataset from 2008 to 2010. The data include time and location information of users' checking-in as is the case in the above datasets. The check-ins are located in various countries.

3. Diffusion-based location-aware recommendation framework

3.1. Time hybrid spreading (THybridS)

The recommender system we use in this study is base on the probabilistic spreading (ProbS) [23]. Two network-based recommender algorithms were developed to perform recommendation, with very different results. The first one, ProbS based on the process of random walk [36] performs recommendation with high accuracy. While the second one, base on Heat diffusion process (HeatS) [37] has low accuracy but tends to favor the diversity of the recommended objects. As the two are based on the physical process of diffusion, they can be merged together in an elegant way [24]. The merging of the two processes results in a method that is at the same time more accurate and more diverse than both processes separately. This is an important feature of this algorithm, as recommender systems tend to be biased towards the popular items [38].

The recommendation score of THybridS is obtained first through propagation and process, and then by adjusting the scores to account for the recent activity of the network. Mathematically, the propagation matrix writes:

$$W_{\alpha\beta}(t) = \frac{1}{k_\alpha(t)^{1-\lambda} k_\beta(t)^\lambda} \sum_{j=1}^{U} \frac{a_{j\alpha}(t) a_{j\beta}(t)}{k_\alpha(t)}, \tag{12}$$

where λ is a parameter between 0 and 1 that tune the hybridization between HeatS ($\lambda = 0$) and ProbS ($\lambda = 1$). The recommendation score of item α for user i is then computed as

$$r_\alpha^{(i)}(t) = \sum_{\beta=1}^{I} W_{\alpha\beta}(t) a_{i\beta}(t). \tag{13}$$

3.2. Users' geographical position

In most datasets, the position of items is given. But the position of users is undefined for privacy concern.

In this work, we make the assumption that each user has a favorite area where he goes regularly. We refer to this area as the *ideal position* of a user. From this

---

[8] https://sites.google.com/site/yangdingqi/home/foursquare-dataset

[9] https://snap.stanford.edu/data/loc-brightkite.html

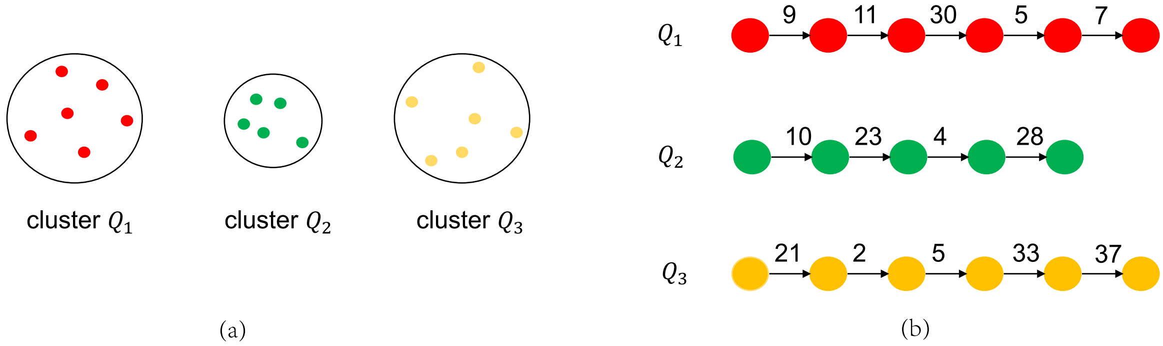


Figure 1. Illustration of DBSCAN clustering algorithm. For a given user i, (a) shows the different candidate clusters of him. (b) Shows the check-in intervals included in the candidate clusters with of which unit is days.

assumption, we can define a user's position as the average position of his business' reviews. However with this approach, there is one major drawback: if a user lives in New York and regularly reviews businesses there, his average position will be located somewhere in New York. But if he goes on vacation two weeks to Paris and reviews business here, his average position will suddenly be somewhere in the middle of the Atlantic ocean.

In order to solve this problem, we use a clustering algorithm [39]. The idea behind a clustering algorithm is to divide items into distinct clusters. Based on the previous example, the clustering algorithm is useful to categorize which reviews of the users are located in New York, and which ones are located in Paris. In order to perform this task we choose a classic and simple algorithm named *DBSCAN* [40, 41] to determine the ideal position of users. There are three adjustable parameters: ϵ as the radius of the cluster. *MinItem*, the threshold for the minimal number of items contained in the cluster. And *MinStep*, the threshold of the time interval for collection in the cluster. This parameter was added to avoid defining a holiday position as an ideal position. For a given user i, the process to determine its ideal position is the following.

- Randomly select an item α collected by user i. If there are at least *minItem* items selected by user i are contained in the circular area with the position of item α as its center and of radius ϵ, this circular area defines a cluster for user i.
- Repeatedly scan all the items collected by user i, and obtain a certain number of candidate clusters Q_1, Q_2, Q_3 as shown in figure 1(a).
- Select the optimal cluster from candidate clusters. As shown in figure 1(b), items in each cluster are arranged according to their collection time, and the time interval of two items is marked on the horizontal line with arrow between two items. Then count the number of items that were collected with a time separation of at least *MinStep* in each cluster. The cluster with the largest number is chosen as the optimal cluster for user i. In our experiment, ϵ is set to 30 km, *MinItem* is 4, and *MinStep* is 20 days. As we can observe, cluster Q_1 contains the largest number of time separations that are no bigger than *MinStep*. So cluster Q_1 is the optimal cluster for user i.

The ideal position of the user is computed as the average position of items in his optimal cluster.

$$\mathbf{x}_u = \frac{1}{|Q_o|} \sum_{\alpha \in \mathcal{Q}_o} \mathbf{x}_\alpha, \tag{14}$$

where Q_o is the optimal cluster of the user i. $\mathbf{x}_i = (x_i, y_i)$ and $\mathbf{x}_\alpha = (x_\alpha, y_\alpha)$ are the vectors of user i position and item α position, respectively, which are composed of latitude x_i/x_α and longitude y_i/y_α . The distance between user i and item α reads:

$$d_{i\alpha} = \sqrt{(x_i - x_\alpha)^2 + (y_i - y_\alpha)^2}. \tag{15}$$

3.3. Geo-THybridS

In the user-driven datasets, the geographical position is important. For instance, restaurants and shops have a physical location. Without geographical information, the recommendation results can be tricky and inept. According to [42], people tend to go to nearby places and periodically visit certain places. This information is quite important, as we can assume that people tend to go regularly in the same areas, because they live or work close to that area, or they are especially attracted to it. Based on this reasonable assumption, we proposed a combination of THybridS algorithm with the geographical position. The calculation formula is as follows.

$$g_\alpha^{(i)} = r_\alpha^{(i)} \exp{(d_{i\alpha}\theta)}, \tag{16}$$

where $d_{i\alpha}$ is the distance between user i and item α, and θ is an adjustable parameter. When θ is negative, the algorithm gives additional weights to items close to the user, while when it is positive it favors distant items.

4. Baselines

4.1. Fusion-algorithm

In order to validate the performance of our algorithm, we compare it with a standard one in the literature that we name *Fusion-algorithm* in this work [42]. This algorithm uses two different scores, one based on users similarity, the other based on geographical position. The scores are calculated separately and then combined together.

The first one computes users similarity, closely related to collaborative filtering [28].

$$\mathrm{sim}_{i\alpha}^{\mathrm{users}} = \frac{\sum_{j=1}^{U} w_{ij} a_{j\alpha}}{\sum_{j=1}^{U} w_{ij}}, \tag{17}$$

where $a_{j\alpha}$ are elements of the adjacency matrix and the cosine similarity w_{ij} is

$$w_{ij} = \frac{\sum_{\alpha=1}^{I} c_{i\alpha} c_{j\alpha}}{\sqrt{\sum_{\alpha=1}^{I} c_{i\alpha}^2} \sqrt{\sum_{\alpha=1}^{I} c_{j\alpha}^2}}. \tag{18}$$

https://doi.org/10.1088/1742-5468/ab74c5

The geographical score of this algorithm is computed as:

$$\mathrm{sim}_{i\alpha}^{\mathrm{geo}} = \prod_{\beta\in\mathcal{I}_i} f(d(\alpha,\beta)), \tag{19}$$

where $\mathcal{I}_i$ is the set of items of user i, and $d(\alpha,\beta)$ the distance between item α and item β. $f(x) = a\exp(bx)$ is a function of the distance, in this work we choose an exponentially decaying function with two parameters a and b. Finally, the two scores are combined together as:

$$\mathrm{sim}_{i\alpha}^{\mathrm{fusion}} = (1-\delta)\mathrm{sim}_{i\alpha}^{\mathrm{users}} + \delta\mathrm{sim}_{i\alpha}^{\mathrm{geo}}, \tag{20}$$

with an additional parameters δ such that $0 \leqslant \delta \leqslant 1$.

4.2. Closest-item

We add the spatial distance in the THybridS, but we also want to consider the performance of the spatial distance on its own as a baseline performance. We design a Closest-item method, which only uses geographic information for recommendation. It ranks items according to the distance between products and users, and then generates a recommendation list. The calculation of the score $f_{i,\alpha}$ for item α and user i is as follows:

$$f_{i,\alpha} = \frac{1}{d_{i,\alpha}}, \tag{21}$$

where $d_{i,\alpha}$ is the distance between user i and item α.

4.3. SVDPP

The third baseline algorithm included in our study is called SVDPP. This is extension of SVD method [43]. A scoring matrix R that includes ratings of items by users can be decomposed into two low-dimensional matrices:

$$R = P^T Q, \tag{22}$$

where $P \in R^{f*U}$ is for user i, and $Q \in R^{f*I}$ is for item α.

SVDPP is the algorithm that adds implicit feedback to SVD [44]. $p_i \in R^f$ is the user-factor vector for user i while $q_\alpha \in R^f$ is the item-factor vector for item α. $r_{i\alpha}$ is the predicted score of user i for item α which is computed as follows:

$$r_{i\alpha} = \mu + b_i + b_\alpha + q_\alpha^T\left(p_i + \frac{1}{\sqrt{|N(i)|}}\sum_{\beta\in N(i)} y_\beta\right), \tag{23}$$

where μ is the global average of scores for all records in the training set, b_i is the user bias and b_α is the item bias. $N(i)$ represents the set of items that have been collected by user i, and y_β is the attribute of the item β. Note that this system predicts ratings rather than predict whether the user will go to a location or not.

5. Results

As mentioned in section 2.1.2, Recall and Precision are both important and complementary metrics to characterize the accuracy of a result. However, Precision is affected by the sparsity of the data: in our data, many users have only one or two new items in the test set. Precision of a recommender system for these users is often very low. However, it does not mean that the recommendation system is ineffective. Thus, the normalization brought by the Recall metric allows more attention to these users. Therefore, all our results are optimized according to the Recall metric. However, note that optimizing with Precision does not change the results significantly. We show the Recall, Precision, and Novelty as a function of the distance parameter θ, and the tuning parameter λ on the three datasets in figure 2. The first interesting result is that the relevant values for θ are around -0.1 for Yelp. In Yelp dataset, we use kilometers for the distance between users and the location of the shops. This means that users typically favor shops within a radius of ten kilometers by a factor of $1/e$, and that around 90% of their interests are located within twenty kilometers radius. There are two regions where Recall is in dark red. This is due to the multiple cities in the dataset, which have different typical distances. We found the highest Recall to be located at around $\theta = -0.1$, and $\lambda = 1.0$. Precision also has two corresponding dark green regions, which indicates that the impact of parameters on Precision is similar to that on Recall. Moreover, Novelty increases with decreasing λ, and reaches its maximum when λ is 0. This is because $1 - \lambda$ represents the proportion of HeatS which provide more contributions to the diversity of recommendation.

For the Foursquare dataset, we see directly that the optimal θ is smaller than for Yelp dataset. This means that the typical activity radius in Tokyo is bigger than in Yelp dataset (Las Vegas and Phoenix cities). This difference might come from the cities, but also from the type of data, as a lot of check-ins in Foursquare are about train stations. This type of data is not present in Yelp, as it consists of business reviews. We also note that above $\theta = -0.075$ the Recall rises again, which can be due to the typical distance between working and living locations. For Foursquare, the optimal λ is 0.9 which indicates that users' choices in the dataset are less driven by popularity than for Yelp.

For the Brightkite dataset, the typical value of θ is around -0.0001, which is much smaller than the typical value of the other datasets. This is due to the fact the distance between the user's location and their check-ins is more broadly distributed compared to the two other datasets. The optimal value of λ is 0.8.

The results for the three datasets of the optimal λ and θ values are shown in table 2. For the three datasets, the Recall is greatly improved: 5.5% for Yelp, 8.7% for Foursquare, and 2.5% for Brightkite. What's more, the results also show that our method consistently outperforms Geo-THyBridS in Precision for all three datasets. Specifically, Geo-THybridS achieves 8.3% improvement over THyBridS on Yelp, 2.6% on Foursquare and 1.8% on Brightkite. The AUC was not improved by adding the geographical modification on all the three datasets. While an improvement of this metric would be ideal, the top of the recommendation list, which is evaluated with Recall is more relevant. With many items in the dataset, the rank of the item does not change

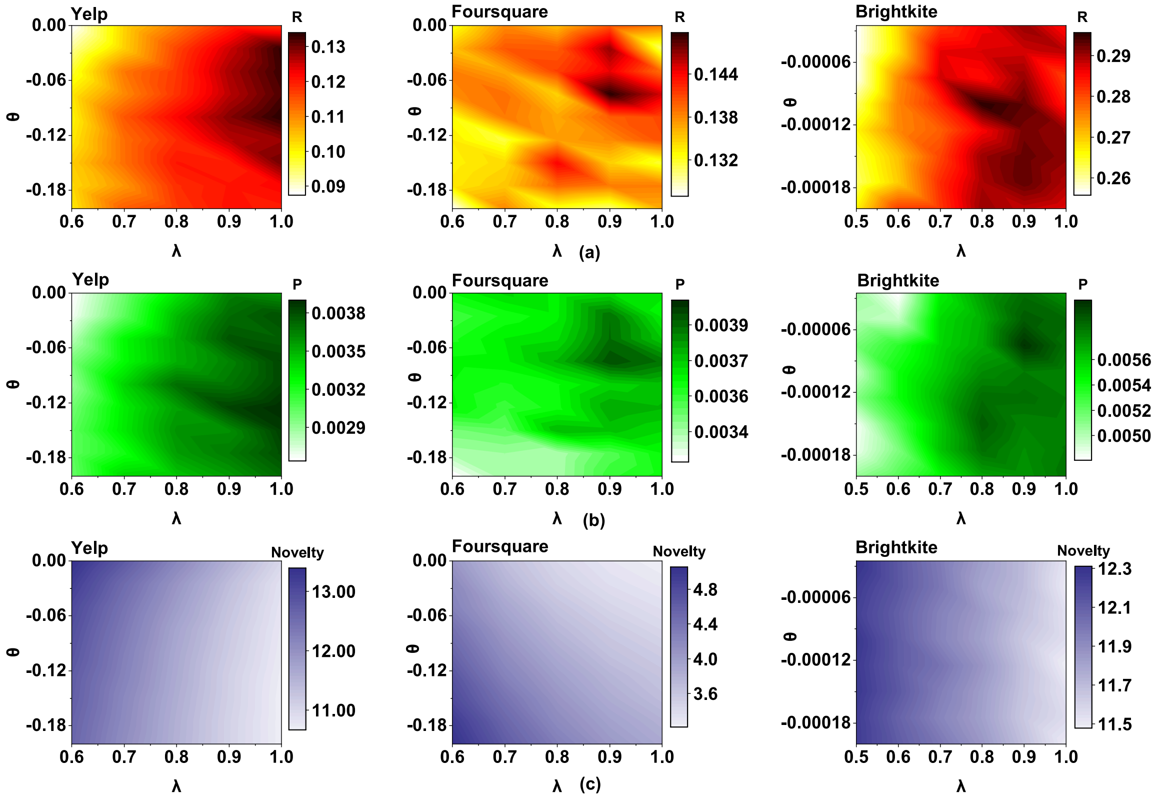


Figure 2. Heatmap of recall, precision and novelty on the three datasets as a function of the distance parameter θ, and the hybrid parameter λ.

significantly the AUC score, for instance, the AUC score would not change much if an item is ranked 20 or 50. But for users, it is quite important.

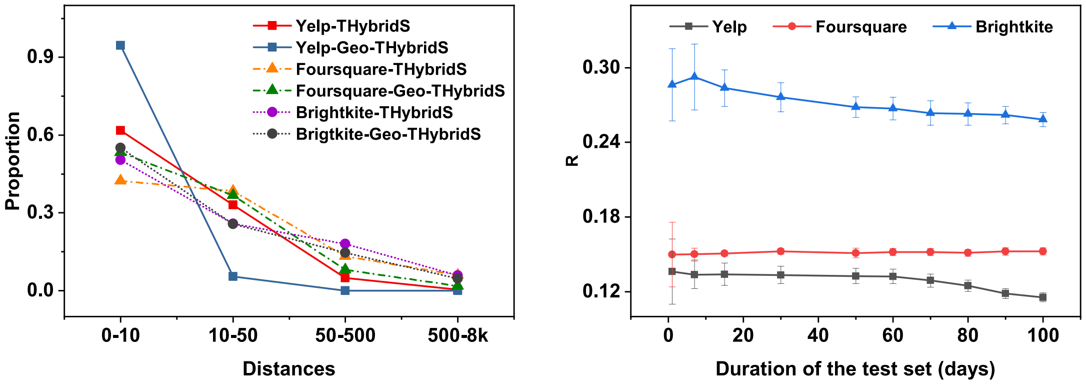
The mean distance of the recommended items was significantly reduced for the three datasets. It is nearly divided by a factor 10 in the Yelp, decreased by around 20% in the Foursquare, and divided by nearly 40% in Brightkite.

In figure 3(a), we show the distance distribution of users to their recommended items. From the distribution, we see that the improvement on the mean distance metric is not due only to some outliers that would be far away from the users with the THybridS algorithm. The improvement is due to the recommended items being closer to users in general. As the result shows, for the Yelp dataset, more than 90% of the items recommended by our algorithm are between 0 and 10 km, and the rest are mostly in the 10–50 km range. For THybridS, only about 60% of the items are in the 0–10 km region, and 33% of them are 10–50 km, and the remaining 5.3% are farther away. For the Foursquare dataset, More than 50% of the recommended items lie in the 0–5 km range, while for ThybridS it is only about 40%. For the remaining larger distances, the distribution of the THybridS is higher than that of Geo-THybridS. For the Brightkite dataset, the distribution difference between the two methods is relatively small, but we still can clearly see the advantages of the algorithm with geographical information. Compared with 50% of the recommended items located between 0 and 30 km for THybridS, Geo-THybridS is 5% higher. For the rest of the larger distances, the

J. Stat. Mech. (2019) 043401

Table 1. Basic information of three used datasets: the number of users U, the number of items I, the number of links L, the average degree of users $\langle k_i \rangle$, and the average degree of items $\langle k_\alpha \rangle$.

	U	I	L	$\langle k_i \rangle$	$\langle k_\alpha \rangle$
Yelp	123 368	41 958	804 789	6.5	19.6
Brightkite	37 303	15 651	201 308	92.3	3.4
Foursquare	2293	61 852	211 670	5.4	12.9



(a) (b)

Figure 3. (a) The distance distribution of users and recommended items. (b) The Recall as a function of the number of test days for the three datasets. The values of the parameters are optimized for the best recall when the number of test days is set to one.

percentage of our method are all slightly less than that of the THybridS. Figure 3(a) indicates that our algorithm encourages users to discover local items.

In our experiments, the test set duration is set to one day, which means it predicts what the users will do for the next day. We change the test day from 1 to 100 and observe the change of Recall. As shown in figure 3(b), the Recall of the three datasets remain stable when the time changes. There is no excessive fluctuation, which shows that the results are not affected by the choice of a specific duration for the test set.

Regarding the diversity metrics, the Novelty for the Foursquare and Brightkite remained stable. Although the Novelty of Geo-ThybridS for the Yelp is lower than THybridS, it is not so much decreased as well. Additionally, the Coverage for the three datasets is also consistent with the THybrid. This shows that our method can keep diversity while improving accuracy.

We compare the Geo-THybridS algorithm with four other baseline methods in figure 4 as a function of the length of the recommendation list N. The parameters for all algorithms were optimized according to the maximization of Recall. The two best performing algorithms for all metrics are THybridS and Geo-THybridS. Two of the other algorithms, Fusion-algorithm and Closest-item, also use geographical data. As the Fusion-algorithm uses a neighborhood-based collaborative filtering, it is a memory intensive algorithm, we had to reduce the network size to 5000 users and 10 000 items, and reduce the number of different samples from 100 to 10. The results are clear and our algorithm clearly outperforms the other methods in terms of accuracy. For the

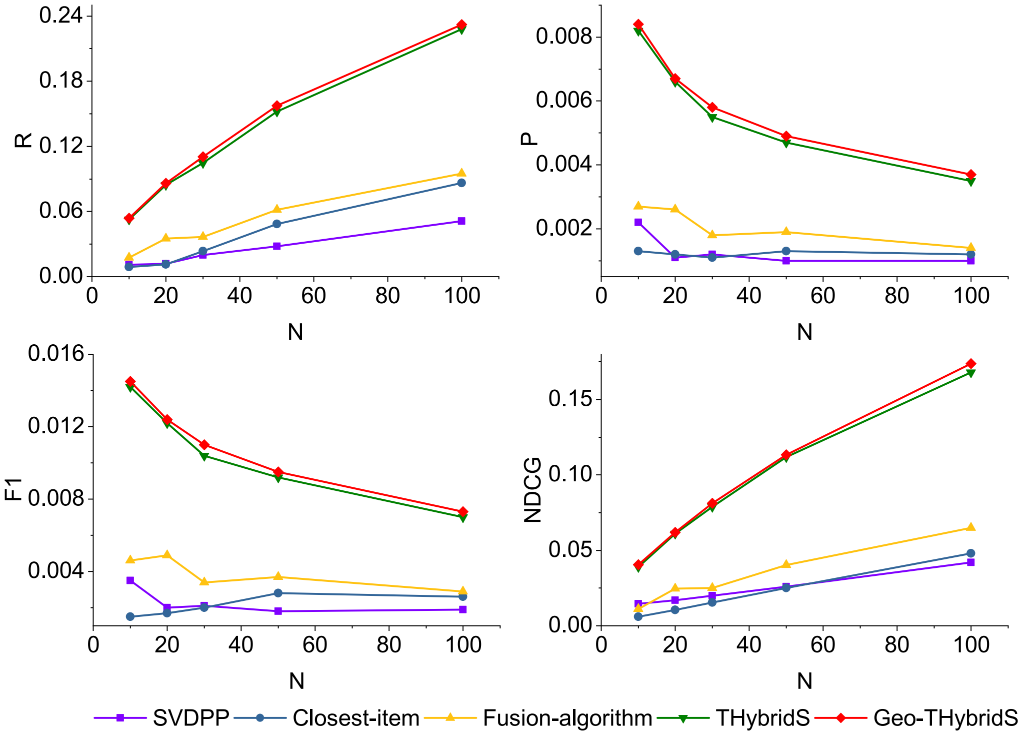


Figure 4. Comparison of the Geo-THybridS algorithm with the other four methods on the Yelp dataset as a function of the length of the recommendation list N.

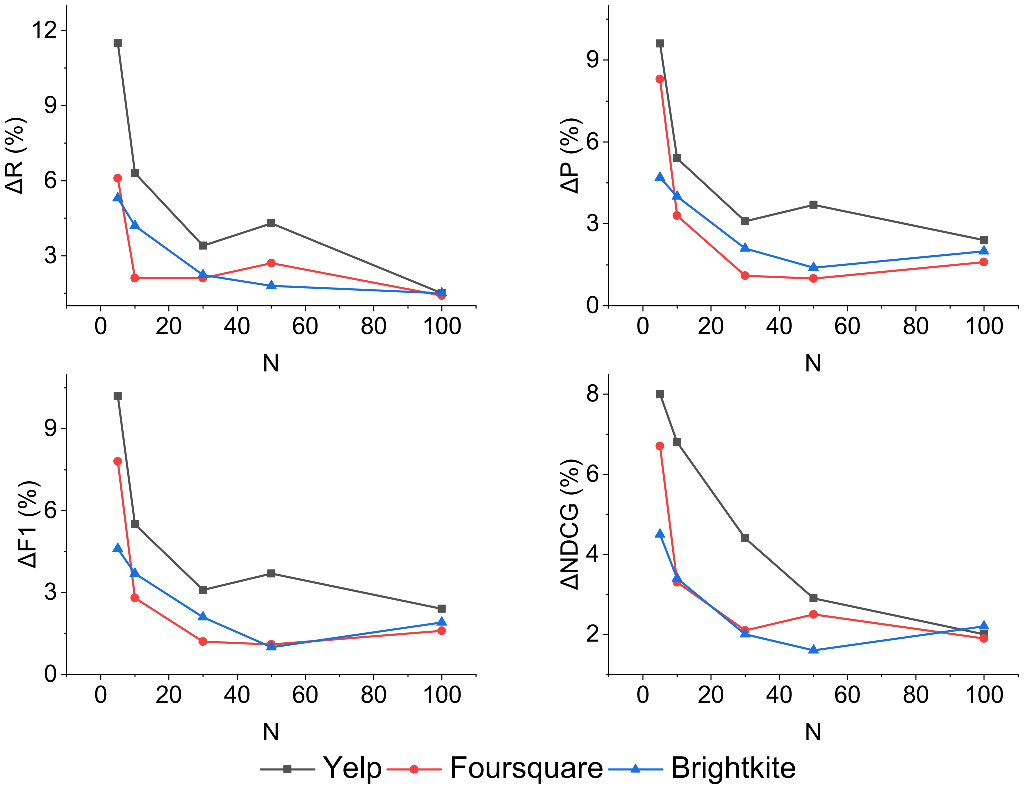


Figure 5. The improvements of the four accuracy metrics for the Geo-THybridS and THybridS algorithms for all three datasets. The duration of the test set is 7 days.

Table 2. Comparison of accuracy, diversity and mean distance of the THybridS and Geo-THybridS on the three datasets. The length of recommended list $N = 50$. The values of the parameters are optimized for the best recall.

		Accuracy				Diversity		
		P	R	AUC	NDGC	Novelty	Coverage	Mean distance
Yelp	THybridS	0.0036	0.127	0.907	0.093	11.4	6.8%	46.39
	Geo-THybridS	**0.0039**	**0.134**	0.896	**0.096**	10.7	6.8%	**5.09**
Foursquare	THybridS	0.0038	0.138	0.888	0.107	4.1	2.6%	7.1
	Geo-THybridS	**0.0039**	**0.150**	0.881	**0.115**	4.1	2.6%	**4.73**
Brightkite	THybridS	0.0055	0.283	0.877	0.172	11.3	1.8%	757.9
	Geo-THybridS	**0.0056**	**0.290**	0.879	**0.179**	11.3	1.8%	**537.1**

Yelp: THybridS with $\lambda = 1.0$; Geo-THybridS with $\lambda = 1.0$ and $\theta = -0.1$.
Foursquare: THybridS with $\lambda = 0.8$; Geo-THybridS with $\lambda = 0.9$ and $\theta = -0.075$.
Brightkite: THybridS with $\lambda = 0.8$; Geo-THybridS with $\lambda = 0.8$ and $\theta = -0.0001$.

Recall metric, our algorithm performs 3.3% better than THybridS as well as outperforming the Fusion algorithm by a factor of three, the Closest-item method by a factor of five, and SVDPP by a factor of ten. Other three accuracy metrics Precision, F1 and NDCG have also been significantly improved by our method.

In order to see clearly the difference between Geo-ThybridS and THybridS, we increase the duration of the test set to seven days and compute the relative improvement for the four accuracy metrics. The results are shown in figure 5. The improvement of Recall, Precision, F1, and NDCG are consistent overall choices for the length of the recommendation list, but stronger when the list is shorter. This indicates that geographical modification improves especially the top of the recommendation list.

6. Conclusion

In this paper, a way of improving the recommendation process using the THybridS recommender system was shown. We demonstrated that it is not required to have the GPS location of the user to improve the recommendation. Indeed, people go out with friends and often at the same places, not necessarily close to their home.

The Geo-ThybridS algorithm recommends objects that are more local orientated. This is a nice feature that promotes local businesses. It is unfortunately not possible for us to test the impact of such a feature, but reducing the distance between users

https://doi.org/10.1088/1742-5468/ab74c5

and their recommended objects could boost the tendency of users to try recommended businesses. We remark that, from a practical view, the possibility of evaluating the impact can be easily developed under our framework when we can obtain local business evolving data.

Given the performance of the THybridS algorithm, an increase of 8% in accuracy is significant. Some additional research directions would be the dependence of the hybridization parameter λ on the location of users. By looking at the shape of the heatmap of Recall on Brightkite dataset in figure 2, it is clear that there are two distinct areas. It is quite possible that one area corresponds to the countryside and small towns, while the other corresponding to large cities. Also, some people tend to go further away to try new restaurants than others. The tuning of the parameter θ according to users' profile may also improve the recommendation process further. Additional insights could be obtained by taking subsets of the Yelp dataset. For example, dividing the datasets into individual cities or districts, and observe the results and parameters of the different locations.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61803266, 61703281, 91846301, 71790615), Guangdong Province Natural Science Foundation (Grant Nos. 2019A1515011173, 2019A1515011064, 2017B030314073), Shenzhen Fundamental Research-general project (JCYJ20190808162601658)

References

[1] Lu L, Medo M, Yeung C H, Zhang C-Y, Zhang Z K and Zhou T 2012 Recommender systems *Phys. Rep.* **519** 1–49

[2] Yu F, Zeng A, Gillard S and Medo M 2016 Network-based recommendation algorithms: a review *Physica* A **452** 192–208

[3] Wu F and Huang Y 2015 Collaborative multi-domain sentiment classification *IEEE Int. Conf. on Data Mining* (IEEE)

[4] Lu Z, Dou Z, Lian J, Xie X and Yang Q 2015 Content-based collaborative filtering for news topic recommendation *29th AAAI Conf. on Artificial Intelligence*

[5] Liao H, Zeng A, Zhou M, Mao R and Wang B-H 2017 Information mining in weighted complex networks with nonlinear rating projection *Commun. Nonlinear Sci. Numer. Simul.* **51** 115–23

[6] Adomavicius G and Tuzhilin A 2011 Context-aware recommender systems *Recommender Systems Handbook* (New York: Springer) pp 217–53

[7] Luca M 2011 Reviews, reputation, and revenue: the case of yelp.com *Harvard Business School Working Paper* (12-016)

[8] Zhou M-Y, Zhuo Z, Liao H, Fu Z-Q and Cai S-M 2015 Enhancing speed of pinning synchronizability: low-degree nodes with high feedback gains *Sci. Rep.* **5** 17459

[9] Wang X, Chen Y, Yang J, Wu L, Wu Z and Xie X 2018 A reinforcement learning framework for explainable recommendation *IEEE Int. Conf. on Data Mining* (IEEE)

[10] Bobadilla J, Ortega F, Hernando A and Gutiérrez A 2013 Recommender systems survey *Knowl.-Based Syst.* **46** 109–32

[11] Said A, Bellogín A, Lin J and Vries A D 2014 Do recommendations matter?: news recommendation in real life *Companion Publication of the Acm Conf. on Computer Supported Cooperative Work and Social Computing* (ACM)

[12] Liao H, Mariani M S, Medo M, Zhang Y-C and Zhou M-Y 2017 Ranking in evolving complex networks *Phys. Rep.* **689** 1–54

[13] Colorni A *et al* 1992 Distributed optimization by ant colonies *Proc. of the 1st European Conf. on Artificial Life* vol 42 (Cambridge, MA)

[14] Dorigo M *et al* 1996 Ant system: optimization by a colony of cooperating agents *IEEE Trans. Syst. Man Cybern.* B **26** 29–41

[15] Li Q, Yu Z, Xing X, Chen Y, Liu W and Ma W Y 2008 Mining user similarity based on location history *ACM Sigspatial Int. Conf. on Advances in Geographic Information Systems*

[16] Ying C and Xu T 2008 Design, analysis, and implementation of a large-scale real-time location-based information sharing system *Int. Conf. on Mobile Systems*

[17] Nan L and Chen G 2009 Analysis of a location-based social network *Int. Conf. on Computational Science and Engineering*

[18] Lian D, Xie X, Zhang F, Yuan N J, Zhou T, Rui Y and Data B 2015 Mining location-based social networks: a predictive perspective *IEEE Data Eng. Bull.* **38** 35–46

[19] Ye M, Yin P and Lee W-C 2010 Location recommendation for location-based social networks *Proc. of the 18th SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*

[20] Liu Y, Wei W, Sun A and Miao C 2014 Exploiting geographical neighborhood characteristics for location recommendation *Proc. of the 23rd ACM Int. Conf. on Information and Knowledge Management* (ACM)

[21] Liu Y, Pham T-A N, Cong G and Yuan Q 2017 An experimental evaluation of point-of-interest recommendation in location-based social networks *Proc. VLDB Endowment* **10** 1010–21

[22] Wang H, Terrovitis M and Mamoulis N 2013 Location recommendation in location-based social networks using user check-in data *Proc. of the 21st ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*

[23] Zhou T, Ren J, Medo M and Zhang Y-C 2007 Bipartite network projection and personal recommendation *Phys. Rev.* E **76** 046115

[24] Zhou T, Kuscsik Z, Liu J-G, Medo M, Wakeling J R and Zhang Y-C 2010 Solving the apparent diversity-accuracy dilemma of recommender systems *Proc. Natl Acad. Sci.* **107** 4511–5

[25] Vidmer A and Medo M 2016 The essential role of time in network-based recommendation *Europhys. Lett.* **116** 30007

[26] Yuan Q, Cong G, Ma Z, Sun A and Thalmann N M 2013 Time-aware point-of-interest recommendation *Proc. of the 36th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval* (ACM)

[27] Billsus D and Pazzani M J 1998 Learning collaborative information filters *ICML* vol **98** pp 46–54

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

[29] Sarwar B *et al* 2000 Analysis of recommendation algorithms for e-commerce *EC* pp 158–67

[30] Hanley J A and McNeil B J 1982 The meaning and use of the area under a receiver operating characteristic (roc) curve *Radiology* **143** 29–36

[31] Zhou T, Su R-Q, Liu R-R, Jiang L-L, Wang B-H and Zhang Y-C 2009 Accurate and diverse recommendations via eliminating redundant correlations *New J. Phys.* **11** 123008

[32] Järvelin K and Kekäläinen J 2000 Ir evaluation methods for retrieving highly relevant documents *Proc. of the 23rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*

[33] Zhou T, Jiang L-L, Su R-Q and Zhang Y-C 2008 Effect of initial configuration on network-based recommendation *Europhys. Lett.* **81** 58004

[34] Ge M, Delgado-Battenfeld C and Jannach D 2010 Beyond accuracy: evaluating recommender systems by coverage and serendipity *Proc. of the 4th ACM Conf. on Recommender Systems* (ACM)

[35] Shani G and Gunawardana A 2011 Evaluating recommendation systems *Recommender Systems Handbook* (New York: Springer) pp 257–97

[36] Fouss F, Pirotte A, Renders J M and Saerens M 2007 Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation *IEEE Trans. Knowl. Data Eng.* **19** 355–69

[37] Zhang Y-C, Blattner M and Yu Y-K 2007 Heat conduction process on community networks as a recommendation model *Phys. Rev. Lett.* **99** 154301

[38] Jannach D, Lerche L, Kamehkhosh I and Jugovac M 2015 What recommenders recommend: an analysis of recommendation biases and possible countermeasures *User Model. User-Adapt. Interact.* **25** 427–91

[39] Jain A K, Murty M N and Flynn P J 1999 Data clustering: a review *ACM Comput. Surv.* **31** 264–323

[40] Ester M *et al* 1996 A density-based algorithm for discovering clusters in large spatial databases with noise *Proc. of the 2nd ACM Int. Conf. on Knowledge Discovery and Data Mining*

[41] Tran T N, Drab K and Daszykowski M 2013 Revised dbscan algorithm to cluster data with dense adjacent clusters *Chemometr. Intell. Lab. Syst.* **120** 92–6

[42] Ye M, Yin P, Lee W-C and Lee D-L 2011 Exploiting geographical influence for collaborative point-of-interest recommendation *Proc. of the 34th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval* (ACM)

[43] Koren Y 2008 Factorization meets the neighborhood: a multifaceted collaborative filtering model *Proc. of the 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining* (ACM) pp 426–34

[44] Kumar R, Verma B K and Rastogi S S 2014 Social popularity based svd + + recommender system *Int. J. Comput. Appl.* **87** 33–7