Vendor-Based Privacy-Preserving POI Recommendation Network

Longyin Cui[1(✉)], Xiwei Wang[2], and Jun Zhang[1]

[1] University of Kentucky, Lexington, KY, USA
lcu225@uky.edu, jzhang@cs.uky.edu

[2] Northeastern Illinois University, Chicago, IL, USA
xwang9@neiu.edu

Abstract. Point-of-interest (POI) recommendation services are growing in popularity due to the choice overloading and overwhelming information in modern life. However, frequent data leakage and hacking attacks are reducing people's confidence. The awareness of privacy issues is multiplying among both the customers and service providers. This paper proposes a localized POI recommendation scheme combined with clustering techniques and introduces the concept of "virtual users" to protect user privacy without sacrificing too much accuracy.

Keywords: Recommender system · Privacy-preserving · Virtual user · Recommendation network

1 Introduction

The demand for Point of Interest (POI) recommendation services is proliferating. Location-based Social Networks (LBSN) providers, such as Yelp and Google Local have effectively increased their market shares. According to Yelp's Q4'19 report, there are 36 million unique mobile app users bringing in revenues over one billion dollars in 2019. Moreover, the total number of user reviews it collected since 2004 has surpassed 205 million [19]. From the Newzoo's Global Mobile Market Report 2020: (1), there are 3.5 billion smartphone users worldwide by the end of 2020; (2), in 2019, about 56% of the global website traffic was generated by mobile devices [13]. Figure 1 shows the primary structure and components of a typical LBSN, in which the records of check-in activities are usually used to generate recommendations.

Conventionally, the data collected is saved and stored in a central server. Such centralized recommender systems are vulnerable when facing data breach issues, and the cost or penalty is substantial. The Capital One data breach, which caused approximately 500 million dollars financial damage on top of other indirect costs [9]. The case study on this incident shows that, nowadays, companies worldwide are not yet adequately adapted to securing their cloud computing environments [14].

J. Xiong et al. (Eds.): MobiMedia 2021, LNICST 394, pp. 477–490, 2021.
https://doi.org/10.1007/978-3-030-89814-4_34

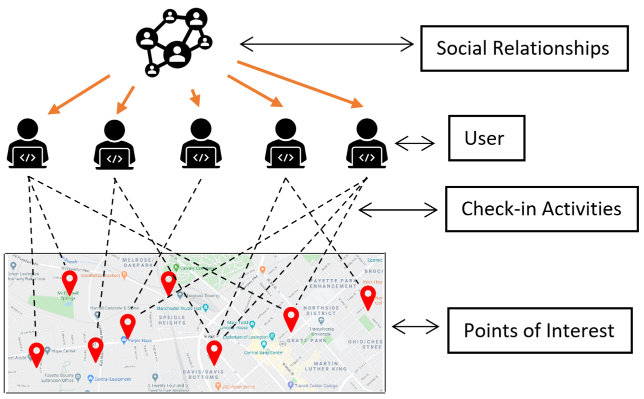


Fig. 1. Location-based social networks (LBSN).

In addition to improving compliance controls, another solution is to push the data processing task to the user end, securing their privacy by eliminating the need for a central server [3,12,17,20]. However, data itself is an essential resource, and the path of giving up storing user information to avoid legal fallouts could be a blind alley. Moreover, these frameworks still require private data, such as social network data or real-time GPS locations. Fetching such data is risky, even under the presence of a privacy disclaimer. In 2019, the Federal Trade Commission issued a 5 billion dollar fine on Facebook due to its violation on consumers' privacy [5].

In this paper, We propose a vendor-based recommendation network scheme. Instead of maintaining a central server or carrying out all computing activities on user sides, we localize the recommendation tasks on vendors for each small area. The vendor can be any Point-of-Interest, such as restaurants, gas stations, and grocery stores. Challenges arise when small business owners decide to build their recommender systems, such as *cold start problem* and *sparsity problem*. Due to the lack of correlation in the high dimensional space, predictions directly made on a sparse feedback matrix often suffer from low accuracy. We alleviated this problem by introducing '***virtual users***', a concept elaborated in later sections that can implicitly reflect real users' preferences.

Both recommendation providers and consumers benefit from this scheme. For service providers, their users are more likely to use the services because local businesses are more trusted than large corporations [16]. For users, their data can be analyzed more efficiently and the risk of having all data hacked at once is remarkably reduced. The customers' chosen vendors also reflect users' active visiting areas making the system naturally location-aware.

Our main contributions are summarized as follows:

– We propose a localized POI recommender system framework, where the computing tasks and data storage of a central server are distributed to smaller areas.

- We introduce the idea of "virtual users", a concept that enables localized recommender systems to collaborate with each other resolving the sparsity problem.
- We conduct experiments on real-world datasets, demonstrating the importance of geographical restriction and the effectiveness of our framework.

The rest of this paper is organized as follows: the background and related works are introduced in Sect. 2. The problem description and our proposed solution are discussed in Sect. 3. Next, in Sect. 4, the experiments are carried out, and the results are analyzed. Section 5 gives the conclusion and future work.

2 Background

2.1 Centralized Recommendations

The major difference between a centralized RS and a distributed or decentralized RS is how data is attained and processed. The data is stored on a single server for a centralized RS where new recommendations are generated immediately after data pre-processing. There are many ways to implement centralized POI recommendation models. To investigate the tradeoff between privacy preservation and recommendation accuracy, we selected several straightforward models to demonstrate the proposed framework. With that being said, the Recommender System Network (RSN) we propose is compatible with various methods.

For example, for a classic Matrix Factorization (MF) model, a regression technique is realized to collaboratively learn the latent factors of users and items (i.e., POIs) [15]. While users' feedback is reflected by their ratings, the latent factors indicate each user or item's hidden characteristics. A general MF based recommendation model can be represented by the following optimization problem as shown in Eqs. (1) and (2), where $\boldsymbol{r}_{ui}$ denotes the known rating given by user $\boldsymbol{u}$ to item $\boldsymbol{i}$, and $\hat{\boldsymbol{r}}_{ui}$ denotes the predicted rating. Vectors $\boldsymbol{p}_u$ and $\boldsymbol{q}_i$ represent the user and item latent factors, respectively.

$$\min_{p_u, q_i} \sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda(\|q_i\|^2 + \|p_u\|^2) \tag{1}$$

$$\hat{r}_{ui} = q_i^T p_u \tag{2}$$

In the well-known biased MF model [11,15], the predicted rating, however, is formalized as follows:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{3}$$

where $\boldsymbol{\mu}$, $\boldsymbol{b}_u$, and $\boldsymbol{b}_i$ represent the global mean, the user bias, and the item bias, respectively. $\boldsymbol{R}_{train}$ is the set of observed ratings. Accordingly, the objective function is then updated as Eq. (4). The notations and the parameters will be discussed in detail in later sections.

$$\min_{p_u, q_i} \sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda(b_u^2 + b_i^2 + \|q_i\|^2 + \|p_u\|^2) \tag{4}$$

We involve Biased MF heavily in our experiment due to its excellent combination of simplicity and reliability.

2.2 Decentralized Recommendations

To convert a centralized RS into a decentralized RS, service providers need to push the data storage and processing tasks to the users' end. Either the user data can be distributed efficiently, or a secure protocol such as a safe peer-to-peer structure is provided to allow information exchange. In a decentralized RS, every user keeps a fraction of the training data and is responsible for generating their own recommendations locally. Some researchers managed to shift the learning process to the users' end to resolve privacy concerns [3,17,18].

However, there are inevitable vulnerabilities in these models. For example, when users exchange ratings directly, a malicious user is able to gather other users' ratings by giving positive feedback to all locations. Alternatively, when only latent factors are exchanged, a malicious user can tell that another user visited a specific place if they share a similar latent factor associated with the same location. Each of the researchers made their breakthroughs and have solved different problems, but many of them remain.

3 Model and Methodology

3.1 Preliminaries

In a centralized or traditional recommender system, suppose we use $\boldsymbol{u}$ to denote a user (customer) and $\boldsymbol{i}$ an item (POI), then $\boldsymbol{U}$ and $\boldsymbol{I}$ are the user and item sets, where we have $\boldsymbol{u} \in \boldsymbol{U}$ and $\boldsymbol{i} \in \boldsymbol{I}$. $\boldsymbol{m}$ and $\boldsymbol{n}$ represent the sizes of $\boldsymbol{U}$ and $\boldsymbol{I}$, respectively. A rating $\boldsymbol{r}_{ui}$ indicates the preference of user $\boldsymbol{u}$ over item/POI $\boldsymbol{i}$. In our dataset, each rating $\boldsymbol{r}_{ui} \in [1, 5]$, where 1 indicates least favored and 5 most favored. As it was introduced in the previous section, we use $\hat{\boldsymbol{r}}_{ui}$ for predicted ratings and $\boldsymbol{r}_{ui}$ for their observed counterparts. Aside from the objective function in Eq. (4), if we denote the rating matrix by $\boldsymbol{R}$, then we have the following formula:

$$R_{m\times n} \approx P_{m\times k} \cdot Q_{n\times k}^T \tag{5}$$

where $\boldsymbol{k}$ is the number of latent factors that are retained, $\boldsymbol{p}_u$ and $\boldsymbol{q}_i$ are column vectors of the two matrices, respectively. For the MF models, unless specified, we use $\boldsymbol{P} \in \mathbb{R}^{m\times k}$ to denote user latent factor matrix, and $\boldsymbol{Q} \in \mathbb{R}^{n\times k}$ to denote the item latent factor matrix. Furthermore, we define $\boldsymbol{T}_r$ as the training set and $\boldsymbol{T}_e$ as the test set. Typically, all MF methods require learning user and item's latent factors by regressing over the known user-item ratings from the pre-processed training dataset [2]. Because of this, both Eqs. (4) and (5) aim to find the optimal $\boldsymbol{P}$ and $\boldsymbol{Q}$ that minimizes $\left\|R - P \times Q^T\right\|$. Finally, we denote the constant in the regularizing terms in Equations such as (1) and (4) by $\boldsymbol{\lambda}$. Both $\boldsymbol{k}$ and $\boldsymbol{\lambda}$ are adjusted and tuned using cross-validation. We use Stochastic Gradient Descent (SGD) to solve the least squares optimization.

In our proposed RSN framework, we break down the centralized RS into multiple local entities. Each area of the city has an independent RS which maintains its own users, and is considered as a local group, denoted by $\boldsymbol{g}_i$ ($\boldsymbol{g}_1 \cup \boldsymbol{g}_2 \cdots \cup \boldsymbol{g}_n = \boldsymbol{U}$). The set of all groups is represented by $\boldsymbol{G}$. In addition to the physical user (real customers) set $\boldsymbol{U}$, we introduce a virtual user (generated fake customers) set $\boldsymbol{V}$. If we define the matrix that stores the virtual users' ratings as $\boldsymbol{R}_v$ and the real users $\boldsymbol{R}_r$, then we have:

$$R_{train} = \begin{bmatrix} R_r \\ R_v \end{bmatrix} \tag{6}$$

In practice, users are encouraged to choose a nearby vendor they trust to receive recommendations. In order to simulate the real scenario in the experiment, users in the dataset are clustered beforehand. The clustering is based on the Pearson Correlation Coefficient (PCC) of users' ratings. Specifically, for any pair of users $\boldsymbol{a}$ and $\boldsymbol{b}$, the similarity between the two is defined as:

$$S_{ab\_PCC} = \frac{\sum_{i \in I_{ab}} (r_{ai} - \mu_a) \cdot (r_{bi} - \mu_b)}{\sqrt{\sum_{i \in I_{ab}} (r_{ai} - \mu_a)^2} \cdot \sqrt{\sum_{i \in I_{ab}} (r_{bi} - \mu_b)^2}} \tag{7}$$

where $\boldsymbol{\mu}_a$ and $\boldsymbol{\mu}_b$ are the average ratings of users $\boldsymbol{a}$ and $\boldsymbol{b}$, and $\boldsymbol{I}_{ab}$ is the item set that $\boldsymbol{a}$ and $\boldsymbol{b}$ both rated. After the affinity matrix is constructed, we then perform the kernel k-means clustering to minimize their in-cluster variance. The PCC is chosen since it has the best performance with respect to mean absolute error (MAE) in neighborhood based RS models [4]. Once all the users are clustered, the cluster centroids are treated as virtual users and sent to all the other RSs.

3.2 Problem Description

We chose two separate cities and their nearby districts to evaluate our model. Most users are only active in a particular town and remain in specific places. This phenomenon is usually referred to as "location aggregation" [3]. For example, Fig. 2 shows the points are aggregated where each point is a visiting record. The x-axis and y-axis represent the user and POI IDs, respectively.

Users select local businesses they trust to share their data before getting recommendation services. Accordingly, we split all the users into different groups to simulate real user activities. This step in simulation is not required in practice since users choose their trusted vendors spontaneously. The paper estimates a user's active location (i.e., the latitude and longitude) by their previously visited POIs (Eq. 8 and 9):

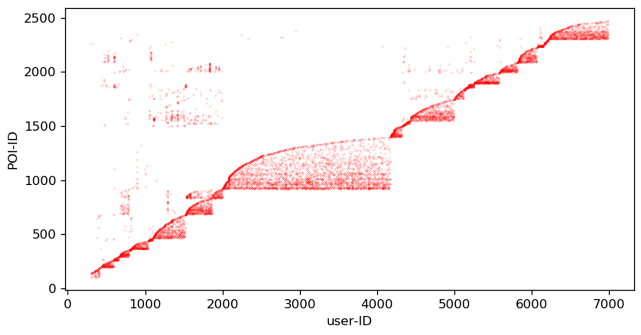


Fig. 2. Yelp dataset user visiting behavior analysis. Each point in the figure represents a check-in record.

$$Lat_u = \frac{1}{|I_u|} \sum_{i \in I_u} Lat_i \tag{8}$$

$$Lon_u = \frac{1}{|I_u|} \sum_{i \in I_u} Lon_i \tag{9}$$

where $\boldsymbol{I}_u$ denotes all the POIs a user visited.

However, the side effect of such action is what it makes the already sparse POI rating matrix even sparser. To increase the number of ratings that can be used to train each model, we provide each local RS with virtual users' ratings. A local RS generates a certain number of virtual users by clustering the existing users. We denote the cluster as $\boldsymbol{c}$ and the user set of that cluster as $\boldsymbol{U}_c$.

$$r_{vi} = \frac{\sum_{i \in I_{uv}, u \in U_c} r_{ui}}{|U_c|} \tag{10}$$

Equation 10 shows how a virtual user rating is estimated. For a virtual user $\boldsymbol{v}$, its rating toward location $\boldsymbol{i}$ is approximated by computing the mean value of the ratings left by users in the same cluster ($\boldsymbol{u} \in \boldsymbol{U}_c$) who visited the same location ($\boldsymbol{i} \in \boldsymbol{I}_{uv}$). The virtual users, whose ratings are shared by all RSs in a RSN, summarize the preferences of physical users.

3.3 Generating Recommendations

When acquiring recommendations, a user downloads the two factorized matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ as defined in Eq. (5), from the local recommender system. This way, the dimensions of the original rating matrix are indirectly reduced, decreasing the download time. The original ratings are also slightly perturbed, making it harder to backtrace the user's checking-in records. The user then reconstructs the rating matrix $\hat{\boldsymbol{R}}$ on his or her personal device. By searching and finding the most similar user, according to (7), the user can acquire all the predicted ratings for any unvisited POIs. Users can choose whether to share the personal information they hold.

The information downloaded is not only anonymous but also a combination of real users and virtual users. The reconstructed matrix $\hat{\boldsymbol{R}}$ is different from the rating matrix stored on the local server since the number of virtual users is dynamic, and all ratings are perturbed.

3.4 The Algorithm

Since the simulation of users choosing trustworthy vendors and generating virtual users play an essential role in our experiment, we organize the work and show it in *Algorithm 1*.

Algorithm 1: Preprocessing User Ratings

```
aInput: all ratings from $R$, all POIs' location information (longitude and
        latitude)
Output: $n$ groups of processed training sets $\{Tr_1, Tr_2, ... Tr_n\}$and test sets$\{$
        $Te_1,Te_2, ... Te_n$ $\}$
1  for $u = 1$ to $U$ do
2      Calculate each user's longitude $Long_u$ and latitude $Lat_u$ according to (8)
       and (9)
3      Eliminate users outside target city
4      Perform $k$-means clustering based on Euclidean distance among users
5  end
6  for $g = 1$ to $G$ do
7      Split $R_g$ into training $Tr_g$ set and test set $Te_g$
8      for user $u$ in $Tr_g$ do
9          Calculate the similarities to all other users according to (7)
10     end
11     Complete affinity matrix for the current group.
12     Perform the kernel $k$-means clustering to generate virtual user ratings $Rv_g$
       according to (10)
13     Append $Rv_g$ to $Rv$
14 end
15 for $g = 1$ to $G$ do
16     Update $Tr_g$ by appending $Rv$ according to (6)
17 end
18 return $Tr$ and $Te$
```

Each vendor-based RS possesses a training set $\boldsymbol{Tr}_g$ in practice, learns the factorized matrices according to (4) and (5), and then make the matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ ready for download for its users. The final recommendations are generated on every user's personal device which further decreases the workload for each vendor-based RS.

3.5 System Update and Maintenance

While the data is static in our simulation, the real-world users continuously move and change their active visiting areas. Furthermore, for privacy concerns, there should not exist a link between cluster centroids and physical users, which prevents the updating process from using the same users. Therefore, each time a clustering is completed, its components, centroids, and the number of clusters will be different from the previous one. To make the cluster centroids better represent real users' personal preferences, two mechanisms are implemented:

- The clustering needs to be regularly performed to generate new virtual users.
- The old virtual users need to be turned inactive after the clustering becomes obsolete.

In real-world scenarios, each virtual user is attached with a timestamp. Once it reduces to zero, the virtual user expires and is then removed. When a virtual user is created, the local RS will broadcast it to all the RSs in the same network. The recipients will decide if the information is useful, depending on the overlap between virtual users' visited locations and the item set of the current RS.

4 Experiments

4.1 Datasets

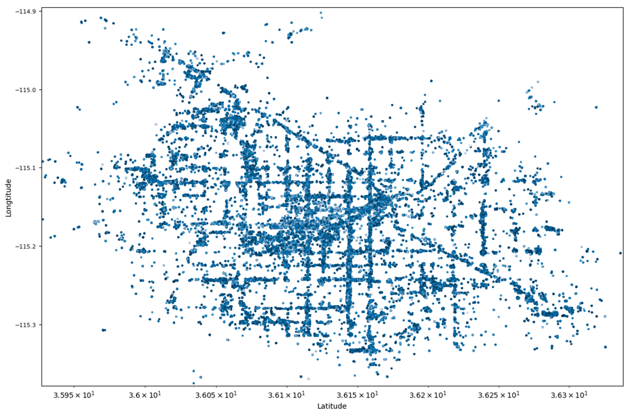


Fig. 3. The user visiting location plots (Las Vegas). Each point represents a user's visiting at real-world geographical position.

We use two subsets of the Yelp business review dataset [1]. The first set was collected in the Urbana-Champaign area, and the second was from the city of Las Vegas and its surrounding areas. The ratings' type is explicit rating (from 1 star to 5 stars) collected by Yelp between January 2007 and December 2017. We removed the users with too few ratings and repeated ratings.

In our test, all local RSs are in the same city or metropolitan area. However, in real-world scenario, if the RSs share any identical items or have overlapped item sets, communication can be established, and RSs in the same RNS can then enhance each other.

During pre-processing, we adopted multiple ways to test the appropriate number of RSs in a network. As discussed in previous sections, we need to group the users based on their visiting locations to simulate the real-world scenario. Figure 3 shows the users' visited POIs, which almost reflects the streets' shape in Las Vegas and nearby areas. However, when we attempt to guess users' real locations by plotting their Euclidean centers of all the visited places, the results did not illustrate apparent segregation. Clustering methods, including k-means, spectral, and density-based spatial, were all tested, and eventually, we chose k-means for its simplicity and straightforwardness. When comparing different clustering methods, we evaluate both the results of accuracy and the balance of user numbers among each area.

Table 1. Datasets statistics

Dataset area	Numer of users	Number of items	Number of ratings
Urbana-Champaign	2737	1502	22654
Las Vegas	31540	30374	802900

After pre-processing, the details of the datasets are listed in Table 1. We sort all the ratings in chronological order and split them by the ratio of 0.2 with the first 80% of ratings for training and 20% for testing.

4.2 Evaluation Metrics

We adopted two metrics to evaluate the model performance, the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). Although accuracy is not always the best metric to evaluate recommender systems [10], minor accuracy improvements, measured by RMSE or MAE, can still pose significant impacts on the quality of top-k recommendations [8,15]. Equations (10) and (11) show the details of the definition:

$$RMSE = \sqrt{\frac{\sum_{u,i \in Te}(r_{ui} - \hat{r}_{ui})^2}{m}} \tag{11}$$

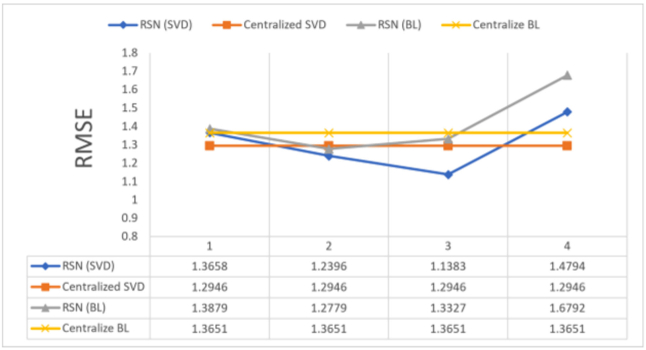
$$MAE = \frac{\sum_{u,i \in Te}|r_{ui} - \hat{r}_{ui}|}{m} \tag{12}$$

4.3 Results and Discussion

In our experiment, we compare our results with three existing models:

– The MF model, promoted by one of the Netflix winners Simon Funk [6]. We chose the one that has integrated the baseline model proposed in [15]. The two latent matrices in (5) are factorized and learned using the objective function in (4). We used a well-built version to represent a typical centralized RS [7].
– The DMF model, a decentralized scheme that only allows users to exchange gradient loss among neighbors during training [3]. Like most decentralized RSs, there is no data stored on the server. All personal information is kept on users' devices.
– The baseline model, of which the prediction function is defined by Eq. 3 but without the last term. A predicted rating is merely calculated by adding the global mean, column bias, and row bias, and there are no iterative updates involved.

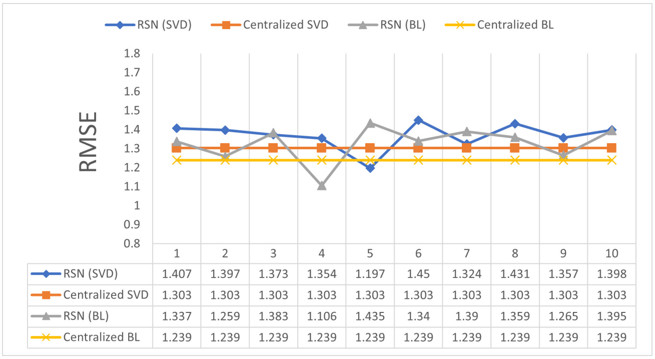
We opt for straightforward recommendation methods over complex models. In our experiment, the core model can be replaced or combined with other schemes. Each local RS can use different algorithms to generate recommendations.



	1	2	3	4
RSN (SVD)	1.3658	1.2396	1.1383	1.4794
Centralized SVD	1.2946	1.2946	1.2946	1.2946
RSN (BL)	1.3879	1.2779	1.3327	1.6792
Centralize BL	1.3651	1.3651	1.3651	1.3651

Fig. 4. Local RSs accuracy results (Urbana-Champaign). Four models' RMSE results for each local RS in Urbana-Champaign area.

In the Urbana-Champaign dataset, users were divided into four sections based on their frequently visited locations. In contrast, the Las Vegas metropolitan area has been categorized into ten smaller regions based on the same criterion. The results of every local RS are shown in Fig. 4 and Fig. 5. The different number of groups is due to the different number of users and the cities' scale. On one hand, if we keep the user size too small for an area, the number of users that can be clustered would be too small, leading to insufficient clusters. On the other hand, if this size is too large, each cluster will have too many users, causing the centroids to be too general to reflect physical users' preferences and interests.



	1	2	3	4	5	6	7	8	9	10
RSN (SVD)	1.407	1.397	1.373	1.354	1.197	1.45	1.324	1.431	1.357	1.398
Centralized SVD	1.303	1.303	1.303	1.303	1.303	1.303	1.303	1.303	1.303	1.303
RSN (BL)	1.337	1.259	1.383	1.106	1.435	1.34	1.39	1.359	1.265	1.395
Centralized BL	1.239	1.239	1.239	1.239	1.239	1.239	1.239	1.239	1.239	1.239

Fig. 5. Local RSs accuracy results (Las Vegas). Four models' RMSE results for each local RS in Las Vegas area.

In Fig. 4 and Fig. 5, for local RSs in RSN, their performance oscillates up and down on the curves formed by centralized RSs. In most cases, their accuracy is only slightly better than each local RS in an RSN. Occasionally, for some specific areas, such as areas 2 and 3 in Fig. 4 or areas 4 and 5 in Fig. 5, RSs in RSN produced higher accuracy than the centralized RSs. In practice, each local RS in an RSN can virtually work with any model and does not have to uniformly use the same method, so theoretically our proposed model has the potential to outperform a centralized RS. The reason is similar to why a hybrid model performs consistently better than a pure model.

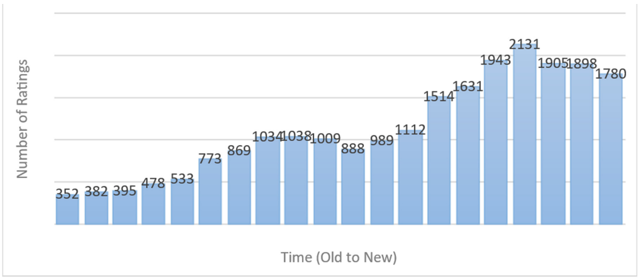
One thing to point out is the way we calculate the average RMSE and MAE. We assume that each region has a local RS to generate its recommendations using real ratings and virtual ratings. It is necessary to estimate the performance of the RSN using all local RSs' average MSE and RMSE. For MAE, the average is the exact mean value of all the MAEs from every local RS. For the RMSE, however, the average RMSE is estimated by calculating the MSE first, and then compute the average RMSE by taking the square root of the mean value of MSE.

As far as hyper-parameters, in Fig. 5, where every RS in the RSN uses biased MF as the default model, the number of latent factors $\boldsymbol{k}(40)$ and learning rate $\boldsymbol{\lambda}(0.1)$ were the same as the centralized MF model. In the DMF model, we used the same value for $\boldsymbol{k}$ and set the regularizer to 0.01 and the learning rates to 0.05. We probed each model with $\boldsymbol{k} \in \{5, 40\}$, the learning rate $\boldsymbol{\lambda} \in \{0.01, 0.5\}$, and the regularizer between $\{0.001, 10\}$.

Table 2 shows the results for different models. It is apparent that all MF models performed better on the Urbana-Champaign dataset. There could be two reasons. First, the Urbana-Champaign dataset is more compact, meaning a small area with relatively sufficient users and POIs to analyze their preferences. Although the Las Vegas dataset is from a densely populated city, it is still too sparse geographically. In fact, this dataset includes visiting records from the city of Las Vegas, North Las Vegas, Spring Valley, Paradise, and all small towns nearby. Second, as shown in Fig. 6, the percentage of new businesses in the Las

Table 2. Datasets statistics

Urbana-Champaign					
Model	MF	RSN(MF)	Baseline	RSN(Baseline)	DMF
RMSE	1.2946	1.3121	1.3650	1.4279	1.4984
MAE	1.0307	1.0568	1.0469	1.0940	1.2018
Las Vegas					
Model	MF	RSN(MF)	Baseline	RSN(Baseline)	DMF
RMSE	1.3034	1.3704	1.2394	1.32996	1.4360
MAE	1.0012	1.1015	0.9452	1.04457	1.1241



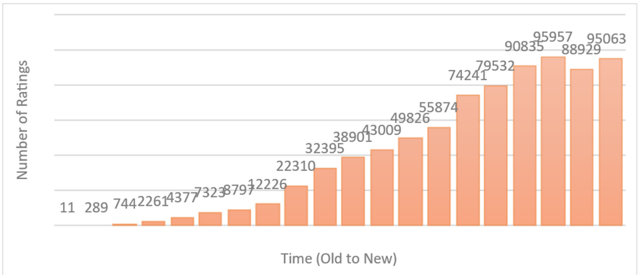


Fig. 6. Rating distributions. (Red: Urbana-Champaign, Blue: Las Vegas). The ratings are ordered from old to new. (Color figure online)

Vegas area is higher than that in Urbana-Champaign. As mentioned previously, we ordered the ratings chronologically, enabling the models to use old data to predict new ratings for simulating real-world scenarios.

In our first dataset, compared to the centralized Biased MF model, the accuracy tradeoff for our RSN model is very small if not trivial (as low as 0.0175 in RMSE and 0.0261 in MAE). The tradeoff is more significant when the recommendation method is changed from Biased MF to the Baseline, but it is still smaller than 0.1. This test result is under the circumstances that all local RSs in our RSN uniformly use the same method and much fewer ratings (1/4 of the total ratings in Urbana-Champaign dataset, 1/10 of the total ratings in Las Vegas dataset). With different recommendation methods implemented, the RSN can achieve better performance. Since each local RS maintains a much smaller set of users to generate recommendations, it reduces the training time. Also, with the help of "virtual users," it scored similar accuracy as a centralized RS. In contrast to a completely decentralized RS such as DMF, the RSN sacrifices much less accuracy for privacy preservation.

In this experiment, we estimated users' preferred vendors who hold their personal history using the locations of their most frequently visited stores. This although is not the most accurate way, is the best option in our assessment due to the limited information. We are positive that in practice, the overall performance of the proposed RSN framework will be better. As a summary, the RSN can lower the privacy risk and boost user confidence with minimal loss of prediction accuracy. Each local RS has a light workload and fast speed of convergence. Moreover, should there be any data breach, it is easier to investigate and control the damage.

5 Conclusion and Future Work

Conventional centralized RSs face high risks in protecting users' privacy because they carry all personal information in the same system. On the other hand, even though decentralized models can eliminate the risks of data breaches, they almost give up all further opportunities for data analysis or mining.

In this paper, we proposed a recommender system network scheme that takes advantage of centralized RSs and decentralized RSs. By introducing virtual users, we can lower the data leakage risk while still generate accurate recommendations. The scheme focuses on both the collaboration among users and among the RSs hosted by small local businesses that people trust.

Future work includes integrating a distributed neural network into our RSN, RS-to-RS communication, and exploring other ways to create virtual or synthetic users, i.e., users who are not physical but can reflect real users' interests and preferences

Acknowledgement. We would like to show our gratitude to the colleagues from the Department of Computer Science at the University of Kentucky and Northeastern Illinois University whose insights and expertise have inspired us. This research was supported by a Committee on Organized Research grant from Northeastern Illinois University.

References

1. Yelp dataset. https://www.yelp.com/dataset
2. Campos, P.G., Díez, F., Cantador, I.: Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. User Model. User-Adapt. Interact. **24**, 67–119 (2013). https://doi.org/10.1007/s11257-012-9136-x
3. Chen, C., Liu, Z., Zhao, P., Zhou, J., Li, X.: Privacy preserving point of interest recommendation using decentralized matrix factorization (2020)

4. Desrosiers, C., Karypis, G.: A comprehensive survey of neighborhood-based recommendation methods. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 107–144. Springer, Boston, MA (2011). https://doi.org/10.1007/978-0-387-85820-3_4
5. Federal_Trade_Commission: FTC imposes $5 billion penalty and sweeping new privacy restrictions on Facebook. Press release 24 (2019)
6. Funk, S.: Netflix update: try this at home (2006)
7. Hug, N.: Surprise: a python library for recommender systems. J. Open Source Softw. **5**, 2174 (2020). https://doi.org/10.21105/joss.02174
8. Koren, Y.: Factor in the neighbors: scalable and accurate collaborative filtering. ACM Trans. Knowl. Discov. Data (TKDD) **4**(1), 1–24 (2010)
9. Lu, J.: Assessing the cost, legal fallout of capital one data breach (2019)
10. McNee, S.M., Riedl, J., Konstan, J.A.: Being accurate is not enough: how accuracy metrics have hurt recommender systems. In: CHI 2006 Extended Abstracts on Human Factors in Computing Systems, pp. 1097–1101 (2006)
11. Mnih, A., Salakhutdinov, R.R.: Probabilistic matrix factorization. In: Advances in Neural Information Processing Systems, vol. 20, pp. 1257–1264 (2007)
12. Nedic, A., Ozdaglar, A.: Distributed subgradient methods for multiagent optimization. IEEE Trans. Autom. Control **54**, 48–61 (2009)
13. Newzoo: Newzoo global mobile market report 2020. https://newzoo.com/insights/trend-reports/newzoo-global-mobile-market-report-2020-free-version/
14. Novaes Neto, N., Madnick, S., de Paula, M.G., Malara Borges, N., et al.: A case study of the capital one data breach. Stuart E. and Moraes G. de Paula, Anchises and Malara Borges, Natasha, A Case Study of the Capital One Data Breach (January 1, 2020) (2020)
15. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2008)
16. SBA_Office_of_Advocacy: Small business profile (2016). https://www.sba.gov/sites/default/files/advocacy/United_States.pdf
17. Wang, X., Nguyen, M., Carr, J., Cui, L., Lim, K.: A group preference based privacy preserving POI recommender system (2020)
18. Yan, F., Sundaram, S., Vishwanathan, S., Qi, Y.: Distributed autonomous online learning: regrets and intrinsic privacy preserving properties. IEEE Trans. Knowl. Data Eng. **25**, 2483–2493 (2012)
19. Yelp: Yelp - company - fast facts (2020). https://www.yelp-press.com/company/fast-facts/default.aspx
20. Yun, H., Yu, H., Hsieh, C., Vishwanathan, S., Dhillon, I.: Nomad: nonlocking, stochastic multimachine algorithm for asynchronous and decentralized matrix completion (2013)