A Unified Framework for User Identification across Online and Offline Data

Tianyi Hao, Jingbo Zhou, Yunsheng Cheng, Longbo Huang and Haishan Wu

Abstract—User identification across multiple datasets has a wide range of applications and there has been an increasing set of research works on this topic during recent years. However, most of existing works focus on user identification with a single input data type, e.g., (I) identifying a user across multiple social networks with online data and (II) detecting a single user from heterogeneous trajectory datasets with offline data. Different from previous works, in this paper, we propose a framework on user identification between online and offline datasets. We build connections between these two types of data by a mapping from IP addresses to physical locations. To solve this problem, we propose a novel framework consisting of three steps. First, we use a clustering method based on locations of IP addresses to map IP addresses into specific physical location distributions. Second, we propose a novel pairwise index to reduce space cost and running time for computing the co-occurrence. Lastly, we apply a learning-to-rank method to merge the effect of multiple features we get in the first two steps. Based on our framework, we design experiments to demonstrate the efficiency (in time and space) of our framework, together with the precision and recall of our approach compared to other methods.

Index Terms—User identification, spatial data mining, heterogeneous data, spatial index.

1 Introduction

During recent years, the usage of mobile devices has been increasing at a tremendous speed, and the online and offline data produced by these devices has drawn a lot of attention of researchers. Online data records various kinds of user activities on websites, such as search engines, social networks and E-commerce sites. Offline data records several kinds of offline activities of users such as offline mobility and check-in data of shops and hotels. There have been some previous works on user identification across multiple sources of online or offline data. For instance, several problems of user identification on online data have been investigated, e.g., in [1], [2], [3], [4], [5], [6], while those on offline data have also been studied, e.g., in [7], [8], [9].

In this paper, we aim to perform user identification across online and offline datasets, instead of working only on one type of data as previous works. The intuition is that, although there exist several kinds of online and offline data, they often share similar features. For online data, since it is generated from online activities, it usually contains IP addresses from mobile devices. For offline data, there is often offline location information of users. By working on them, we identify and merge heterogeneous online and offline data from the same user, and get a set of enriched data for users and a better understanding on their activities.

- *T. Hao and L. Huang are with the Institute for Interdisciplinary Information Sciences, Tsinghua University, China. This work was done when T. Hao was an intern at the Baidu Research. The work of T. Hao and L. Huang is supported in part by the National Natural Science Foundation of China Grant 61672316, the Zhongguancun Haihua Institute for Frontier Information Technology and the Turing AI Institute of Nanjing. E-mail: {haoty14@mails,longbohuang@mail}.tsinghua.edu.cn*
- *J. Zhou is with the Business Intelligence Lab, Baidu Research, and National Engineering Laboratory of Deep Learning Technology and Application, China. E-mail: zhoujingbo@baidu.com*
- *Y. Cheng and H. Wu are with the Big Data Lab, Baidu Research, China. E-mail: {chengyunsheng01,wuhaishan}@baidu.com*
- *J. Zhou and L. Huang are the corrersponding authors.*

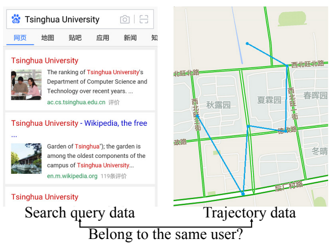


Fig. 1: User identification: linking users from online data and the offline data

The key of our work is to build connections between carefully chosen datasets. Specifically, we choose two datasets with these two representative online and offline features. For online data, we use mobile query log data, which is generated from searching activities on search engines with a mobile browser. From this data, we obtain cookie IDs as the user IDs. For offline data, we use user trajectory data, which is generated from location-based service applications from mobile apps. From this data, we have the mobile device IDs as the user IDs. We then identify and connect different user IDs belonging to the same users across online and offline data. As shown in [10], our results can benefit many user-based applications, such as recommender systems. For instance, if the online shopping data on e-commerce sites and the offline check-in data in restaurants and supermarkets can be merged together for the same users, we can have a better understanding on the behavior of the users and provide more rational recommendations for them. We illustrate a motivation example in Fig. 1.

According to our observation, there exist many IP addresses, which usually appear in several relative stable physical areas, e.g., a campus, a community or a shopping mall. Our intuitive insight for tackling this problem is that for a device ID and a cookie ID, if trajectory points always appear in the areas as IP addresses of search queries, it is high likely that these IDs belong to the same user. In fact, this intuition has rigorous theoretical background that human mobility has high uniqueness [11], [12], [13]. Therefore, computing the co-occurrences of trajectory points in the location distribution areas of IP addresses can help us connect the same user in online and offline data. In addition, the online and offline data generated by the same user may share many common features such as operation systems and mobile phone models. It is also desirable to utilize this information to improve the identification accuracy.

In this paper, based on the above insights, we propose a novel and practical user identification framework between online and offline data, called UIONF, for linking users in heterogeneous online and offline datasets. The flowchart of UIONF is shown in Fig. 2, which consists of three key steps.

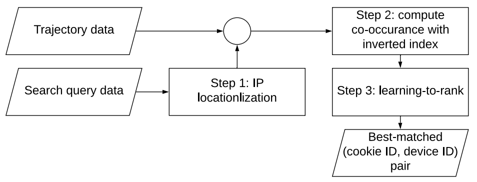

Trajectory data
Search query data
Step 1: IP locationlization
Step 2: compute co-occurance with inverted index
Step 3: learning-to-rank
Best-matched (cookie ID, device ID) pair

Fig. 2: Flowchart of the UIONF framework

The first step is IP localization. We propose a method to differentiate the IP addresses distributed in a set of relative stable and small location areas. After obtaining the IP locations, we can infer several possible locations for each search query, together with the believability of each location. The second step is to compute the co-occurrences between trajectory points and IP locations, for selecting candidate pairs for the ID linkage. We propose a Term Frequency-Inverse Document Frequency (TF-IDF) model based metric [14] to measure the co-occurrence similarity. Since the computation of the co-occurrence similarities has a huge complexity of $O(n^2)$ in both time and space, we design a novel pairwise-index with graph-based optimization algorithm based on the Map-Reduce framework. In this step, for each cookie ID in the search query dataset, we obtain the top-K matched device IDs from the trajectory dataset. Finally, in the last step, for all candidate cookie-device ID pairs, we employ a learning-to-rank (LTR) [15] model on the top-K matched device IDs for each cookie ID, to obtain a finalized best-match pair between a cookie ID and a device ID. This LTR model can utilize all user features shared between online and offline data, including location-based features, e.g., spatial similarity, and other related features, e.g., mobile operation system and mobile phone models.

In summary, our main research contributions include:

- We first study the user identification problem between online and offline data. Specifically, we investigate the problem on mobile query log data and trajectory data, which are typical human behavior data in online and offline scenes, respectively. We design a novel approach by IP-location clustering to build a connection between online and offline data.
- We propose a novel and practical framework, called UIONF, to solve the user identification problem between online and offline data. The framework can efficiently process real-world large datasets on the Map-Reduce platform. In the framework, we invent a novel pairwise index with graph-based optimization algorithm to compute the similarities between pairs of users, by which we can efficiently process a very large dataset in a short time with significantly reduced space cost.
- We apply a learning-to-rank method to generate the best-matched ID pairs. As far as we know, there is no existing works using learning-to-rank method to deal with user identification problem.
- We conduct extensive evaluations on real-world datasets to demonstrate the effectiveness and efficiency of our framework. Our experiments demonstrate that our framework can process the data from megacity like Beijing with more than 10 million user IDs in acceptable time and space cost.

Compared to our earlier work in [16], the improvements include:

- In the IP clustering step, we replace the DBSCAN method in [17] with a method based on the Gaussian mixture model [18], which is a method where the parameters can be fixed more automatically, and can be used to make more accurate prediction.
- For the original inverted index, we propose a novel pairwise-index to increase the efficiency of our framework.
- We add a novel spatial-temporal-based index to make full usage of the time information in our datasets. We use the outputs of this model as new features in the learning-to-rank step.
- We design several experiments to evaluate our framework, and use a much larger dataset from Beijing to show that our framework can efficiently handle large-scale problems.

2 Related Work

This paper is an extension of our previous work [16] on user identification across cyber and physical spaces (UNICORN). We make novel modifications to our original model to improve its effectiveness. In addition, our work is closely related to the topics of user identification and similarity search based on user mobility data. The method for user identification by matching similar user trajectories has been introduced in [9], which uses multi-layer grid indexes in filtering and co-occurrence signals in similarity computation. Authors in [8], [19], [20], [21] have described methods for user identification based on similarity search among trajectories. Our work differs from these works since we use both the online and offline data.

There are also some other researches on the similarity search of user trajectories, such as [22], which proposed the idea of attaining lower bounds of similarities during the similarity searching process. As our algorithm for computing similarities is based on the inverted spatial index, it is also related to works on spatial indexing, such as [23].

Another related topic is user identification within online data. So far, the most popular problem is how to link and identify users from multiple social networks, e.g., [1], [2], [3], [4], [5], [6]. However, the proposed methods cannot be directly extended to our problem and dataset.

There are also some works on lockstep behaviors. In [24], they proposed a framework CopyCatch in order to detect fake page-likes in Facebook for detecting the lockstep behavior of users. The algorithm in this work is based on local clustering and subspace clustering, together with a MapReduce-compatible algorithm to ensure its scalability. The authors in [25] proposed an algorithm to figure out the lockstep behavior in graphs, such that a group of users who follow the same set of users or items in a network. In this work, the technique of singular value decomposition is applied to detect the lockstep behavior. In [26], the authors proposed an algorithm to identify the crowd frauds on the Internet by analyzing the behaviors of the users, in which a non-parametric clustering algorithm is used. In these works, they developed methods to analyze the commonalities of user behaviors, in order to find similar users in social networks, which have some similar ideas to our work. There are several differences between their works [24], [25], [26] and ours. First of all, in the works about lockstep behavior [24], [25], [26], they are more commonly using cluster-based algorithms, while in our work, we concentrate on pairing user IDs by computing the similarity levels of their features. The clustering method is not compatible with our work, since we concentrate on making exact matching across user IDs, instead of finding a group user with similar activities. Second, in their works, they pay more attention to the user-to-item or user-to-user relationships, while in our works we pay more attention to the spatial-temporal features of users. Third, they mainly focus on analyzing the online behaviors of users, while our work aims to build a connection between the online and offline activities of users.

In addition, since our results are based on computing similarities between TF-IDF vectors, the topic for all-pair similarity searching is also related to our work. The authors in [27] have proposed an optimization algorithm for all-pair similarity searching. However, this algorithm is not suitable for distributed systems, which means it is not compatible with large datasets in our work. Metwally et al. [28] have proposed a "V-SMART-Join" framework, which can run on a Map-Reduce system. However, in our problem, the step of matching pairs of IDs in the same inverted index will process too much output data, and simply dropping big indexes in the classical way will cause serious information loss, which has negative influence on the prediction effect.

3 Problem Statement

Our main purpose is to identify corresponding users from two different datasets: the online and offline data. As we have explained, in practice, we will use trajectories as the offline data and mobile query logs as the online data.

Formally, the input data we use in this paper includes:

- Offline data: The trajectory data is defined as a set S_{L} of user trajectories, which can be represented as

$$S_{\mathrm{L}} = \{\langle id_i^{\mathrm{L}}, \{p_{i,1}, \cdots, p_{i,\mathrm{end}}\}\rangle \mid i = 1, 2, \cdots\}.$$

  Here id_i^{L} is the device ID of the i-th user. There is usually one unique ID related to a mobile device. $p_{i,j}$ is the j-th node in the trajectory of id_i^{L}, and each node has the form $p_{i,j} = (x_{i,j}^{\mathrm{L}}, y_{i,j}^{\mathrm{L}}, t_{i,j}^{\mathrm{L}})$, where $x_{i,j}^{\mathrm{L}}, y_{i,j}^{\mathrm{L}}$ are the coordinates, and $t_{i,j}^{\mathrm{L}}$ is the timestamp.
- Online data: The mobile query log data is the collection of query records from the search box of a search engine generated on mobile phones. For online data, we have a collection S_{Q} of all the IDs and the sets of query records related to them:

$$S_{\mathrm{Q}} = \{\langle id_i^{\mathrm{Q}}, \{r_{i,1}, \cdots, r_{i,\mathrm{end}}\}\rangle \mid i = 1, 2, \cdots\}.$$

  Here id_i^{Q} is the cookie ID of the i-th user using the mobile browser. There may be multiple cookie IDs related to the same user since cookie IDs may change. Each record $r_{i,j}$ belongs to one search query sent by user id_i^{Q}, and $r_{i,j} = (\mathrm{IP}_{i,j}^{\mathrm{Q}}, t_{i,j}^{\mathrm{Q}}, s_{i,j}^{\mathrm{Q}})$. $\mathrm{IP}_{i,j}^{\mathrm{Q}}$ is the IP address, and $t_{i,j}^{\mathrm{Q}}$ is the timestamp. $s_{i,j}^{\mathrm{Q}}$ is made up of some extra information, such as the query string, the operation system of the mobile device and the mobile phone model. For some queries, $s_{i,j}^{\mathrm{Q}}$ may also contain the location where the query was sent, i.e. $(x_{i,j}^{\mathrm{Q}}, y_{i,j}^{\mathrm{Q}})$, but different from the trajectory data, this location information may be invalid, since the location is not essential for search queries.

For each pair of user IDs from these two datasets, say id_i^{L} and id_k^{Q}, our goal is to find whether they belong to the same user. To achieve this, we propose a metric to measure the weighted co-occurrence between the spatial and temporal distribution of id_i^{L} and id_k^{Q}. In spite of this, there are still several challenges that are unsolved.

- A large part of query records do not have exact location coordinates, but all the records have detailed IP addresses. Since IP addresses and locations are related, it is necessary to predict the location distributions of the users based on the data of IP addresses.
- There are different features helpful for measuring the co-occurrence of the users, e.g., the weight of location distributions of each IP address, the frequency distribution that one user visits different places and the visiting popularity of all the possible places.
- Since calculating similarities between pairs of IDs takes an $O(n^2)$ time and space complexity, it is necessary to do optimization to reduce the complexity.

Our solution to this problem consist of three parts.

- In Section 4, we show how to model the location distribution of IP addresses and use it to enrich the mobile query data of users.
- In Section 5, we introduce the algorithm to compute the most similar device IDs to particular cookie IDs. In this step, we will use the TF-IDF metric to measure the co-occurrence similarities between users. We invent a technique called the pairwise-index to speed up the computation process, since it would take a lot of time to match up all pairs of IDs and compute the similarities for a large dataset.
- After obtaining several similar device IDs for one cookie ID, in Section 6, we use a learning-to-rank (LTR) approach to considering more features in depth to get a more accurate prediction.

4 Extending IP Addresses to Locations

In each search box query record, several information entries can be found typically, including: (1) Cookie ID of the user, (2) Timestamp when the query was made, (3) IP address which the query was sent from, (4) Extra information, such as location coordinates, operation system and mobile phone model. Despite the fact that there exists a timestamp and an IP address for each search query record, a large part of them is not attached with location coordinates. In order to link the mobile query data and the trajectory data, our idea is to find out the location distribution for each available IP address, and then predict the location where a query was sent by the IP address if there was no location coordinates attached to the query record.

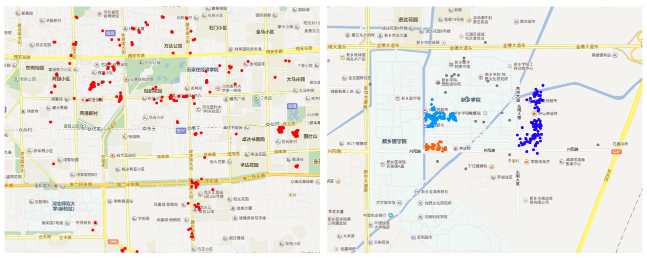

(a) (b)

Fig. 3: Wide-ranged distribution (a) and centralized distribution (b) for IP locations.

After visualizing and analyzing the location distributions of some IP addresses, we find that there are two kinds of distributions for an IP address, as shown in Fig. 3:

- Centralized distribution: The locations of the IP address are distributed in one or several small central areas, the radiuses of which are usually no more than hundreds of meters, and usually only tens of meters.
- Wide-ranged distribution: The locations of the IP address are distributed widely in a large area with a size of several kilometers, and sometimes this area could spread over one or even several cities.

Then it can be figured out that if an IP address follows a centralized distribution, then the locations of the users with this IP address can predicted as the centers of these central areas. To find out IP addresses with a centralized distribution, it is a natural idea to process a clustering on these locations which share the same IP address. In our previous work in [16], we have used the DBSCAN algorithm [17] to process the clustering. In this paper, we use the Gaussian mixture model [18] instead, since it has such advantages:

- Parameters of the model can be better determined with an automatic process.
- It is more convenient to convert a Gaussian distribution into a grid-based distribution.
- According to our experiment, we can get a better predicting accuracy by this method.

According to this, since the centralized distribution usually has a center and a distributing radius, we assume that the distribution of the IP locations follows a 2-dimensional spherical Gaussian mixture distribution, which means, for each IP address, there exist K center points $(\bar{x}_1, \bar{y}_1), \cdots, (\bar{x}_K, \bar{y}_K)$, and for each center point $\boldsymbol{\mu}_i = (\bar{x}_i, \bar{y}_i)$, there is a weight w_i and a variance σ_i^2. Then for each IP address, its location (x, y) follows such a distribution:

$$(x, y) \sim \sum_{i=1}^{K} w_i \mathcal{N}(\boldsymbol{\mu}_i, \sigma_i^2) = \mathcal{G}_K,$$

where $\mathcal{N}(\boldsymbol{\mu}_i, \sigma_i^2)$ is a 2-dimensional spherical Gaussian distribution with the center point of $\boldsymbol{\mu}_i$ and the variance of σ_i^2.

For a fixed center point number of K, with N given samples for this IP address, we can compute the parameters w_i, $\boldsymbol{\mu}_i$ and σ_i of the GMM by the expectation-maximization method [29]. However, since we do not know the number K of clusters a-priori, we still need to fix the value of K before applying the EM-GMM method. In order to decide the value of K, we use the Bayesian information criterion (BIC) [30], which is defined for a distribution $\mathcal{G}_K$ as:

$$\mathrm{BIC}(\mathcal{G}_K) = \frac{1}{2} K \log N - \log p(n_1, \cdots, n_N | \mathcal{G}_K),$$

where $\mathcal{G}_K$ is the Gaussian mixture distribution generated by the EM-GMM algorithm with K components, and $n_1, \cdots, n_N$ are the coordinates of the N samples. The smaller the value of this criterion is, the more likely we will use this distribution. Then, we estimate the value of K as:

$$K^* = \arg \min_{1 \le K \le K_{\max}} \mathrm{BIC}(\mathcal{G}_K),$$

where we set an upper bound $K_{\max}$ for K to prevent the value of K to be too large and that means it is more likely that the IP address distributes as a wide-ranged distribution.

After taking $K = K^*$, we have found the best-matched Gaussian mixture distribution for the locations of the IP addresses. Then we need to convert the IP location distributions into grid distributions in order to continue with the grid-based index algorithm in Section 5.2. Given a distribution $\mathcal{G}_K = \{(\bar{x}_i, \bar{y}_i, w_i, \sigma_i) | 1 \le i \le K\}$ with the parameters fixed, for each square grid $g = (x_\mathrm{a}, x_\mathrm{b}, y_\mathrm{a}, y_\mathrm{b})$, we compute the weight of the IP address in g as:

$$\begin{aligned} w(g) &= \iint_g \mathcal{G}_K(x, y)\, dS \\ &= \int_{x_a}^{x_b} dx \int_{y_a}^{y_b} \sum_{i=1}^{K} w_i \mathcal{N}(\boldsymbol{\mu}_i, \sigma_i^2)(x, y) dy \\ &= \sum_{i=1}^{K} w_i \int_{x_a}^{x_b} \mathcal{N}(\bar{x}_i, \sigma_i^2)(x) dx \int_{y_a}^{y_b} \mathcal{N}(\bar{y}_i, \sigma_i^2)(y) dy, \end{aligned}$$

which is a weighted sum of products of two Gaussian integrals.

According to this formula, we can compute the weights of IP addresses in each grid for a centralized distribution. In order to get rid of the influence of wide-ranged distributions, we ignore grids with a weight of no more than 0.1. That is because, for a wide-ranged distribution, the weights w_i for the center points are usually small, and the variances σ_i is usually large, which brings only a small weight to each grid. That means that we only need to remove all small-weighted grids to exclude the wide-ranged distributions.

Following this approach, for a mobile query record without precise location information, we can predict several possible grids for it by IP clustering, together with the prediction confidence for these locations. In this way, we can successfully build a connection between IP addresses from the online data and location points from the offline data.

5 Computing Similarities

In order to match user IDs from the two datasets, in this section, we propose to use a TF-IDF-based metric to compute weighted co-occurrence similarities between each pair of these cookie IDs and device IDs. The challenge here is that the computation of all user similarities takes a complexity of $O(n^2)$, which brings up a huge computation cost for a large dataset. To tackle this problem, we devise a pairwise index based on the Map-Reduce framework to retrieve the top-K similar device IDs for each cookie ID efficiently.

5.1 Stay Points Extraction

```
1:  function GetStayPoints($T$)
2:      $P_{\text{stay}} \leftarrow \{p_1\}, P_{\text{cur}} \leftarrow \{\}$
3:      for $i = 2 \rightarrow |T|$ do
4:          $v_{i-1} \leftarrow \frac{\sqrt{(x_i-x_{i-1})^2+(y_i-y_{i-1})^2}}{t_i-t_{i-1}}$
5:          if $v_{i-1} < v_{\max}$ then
6:              if $P_{\text{cur}} = \{\}$ then
7:                  $P_{\text{stay}}$.add($p_i$)
8:              else
9:                  $P_{\text{cur}}$.add($p_i$)
10:         else
11:             if $P_{\text{cur}} \neq \{\}$ then
12:                 if $P_{\text{cur}}[P_{\text{cur}}.\text{len}].t - P_{\text{cur}}[1].t \geq t_{\min}$ then
13:                     $P_{\text{stay}}$.concentrate($P_{\text{cur}}$)
14:                 $P_{\text{cur}} \leftarrow \{\}$
15:             $P_{\text{cur}}$.add($p_i$)
16:     $P_{\text{stay}}$.concentrate($P_{\text{cur}}$)
17:     return $P_{\text{stay}}$
```

Fig. 4: Stay points extraction

Although trajectories are important data, they are not appropriate to be used directly. As described in [9], [31], there are usually a large number of moving points and noise points in the trajectories, making less effective information and causing a waste of computation resources. Due to these reasons, an approach to finding stay points has been described in these papers by partitioning the trajectories based on spatial and temporal distances. In our work, since there is no need for trajectory simplification at the stay points, we use an approach different with [9], [31], under which moving or noise points are removed by the speed of users.

Recall that the trajectory T of a user is:

$$T = \{(x_1, y_1, t_1), (x_2, y_2, t_2), \cdots, (x_{\text{end}}, y_{\text{end}}, t_{\text{end}})\},$$

where the nodes $p_i = (x_i, y_i, t_i)$ are sorted in time order such that $t_1 < t_2 < \cdots < t_{\text{end}}$. For the trajectory T, in order to extract stay points, we have the following definitions:

- *Edge*: An edge is a pair of adjacent nodes $e_i = (p_i, p_{i+1})$ for $i = 1, 2, \cdots, |T|-1$. The *time* of an edge is defined as $\Delta t(e_i) = t_{i+1} - t_i$, and the *speed* of an edge is defined as $v(e_i) = \frac{\sqrt{(x_{j+1}-x_j)^2+(y_{j+1}-y_j)^2}}{t_{i+1}-t_i}$.
- *Sequence*: A sequence of nodes is $S_{i,j} = (p_i, p_{i+1}, \cdots, p_j)$, and the *duration* of a sequence is $\Delta t(S_{i,j}) = t_j - t_i$. A sequence $S_{i,j}$ contains several edges $\{e_i, e_{i+1}, \cdots, e_{j-1}\}$.
- *Stationary sequence*: A sequence $S_{i,j}$ is called a stationary sequence iff the speeds of all edges are less than a threshold, i.e., $v_{e_k} < v_{\max}$, for $i \leq k \leq j-1$.
- *Maximum stationary sequence*: A stationary sequence $S_{i,j}$ is called a *maximum stationary sequence* iff there does not exist another stationary sequence $S_{i',j'}$ such that $S_{i',j'} \supset S_{i,j}$, i.e., $i' \leq i, j' \geq j$ and $j'-i' > j-i$.
- *Stay sequence*: A maximum stationary sequence $S_{i,j}$ is called a *stay sequence* iff the duration of this sequence is no shorter than some threshold, i.e., $\Delta_t(S_{i,j}) \geq t_{\min}$, while at the same time this sequence is not at the head or tail of the trajectory, i.e., $i = 1$ or $j = |T|$.

Our goal in this step is to find all nodes that belong to some stay sequences. We start by scanning the speeds of all edges in the trajectory in time order. We use P_{stay} to denote the target set in which we place the stay nodes. We also maintain a sequence P_{cur} to keep track of the current stationary sequence, and update them as follows. After we encounter an "overspeeding edge" on which the travelling speed of the user exceeds $v_{\max}$, we check if the current stationary sequence P_{cur} has a duration of at least $t_{\min}$. If it has, we place all the nodes in the sequence into P_{stay}. Then, we empty P_{cur} and continue with the scanning in the same way, until reaching the end of the trajectory. At last we return the target sequence P_{stay}, which is the set of all stay points. The pseudo-code of the algorithm is in Fig. 4.

Following this algorithm, the moving points and noise points will be placed into the maximum stationary sequences with durations shorter than $t_{\min}$, and will not be in the target sequence P_{stay}.

5.2 TF-IDF Vectors Generating

In our framework, the co-occurrence similarities between trajectories and mobile query records are different from the similarity computations between trajectories in [8], [32]. In our case, we compute the similarities between spatial and temporal distributions, rather than that between trajectory sequences. The main reason is that our input data comes from two different resources. As explained in Section 4, there is no exact location information for some of the query records, and we use the IP location distributions instead. This brings difficulty in building exact trajectory sequences for users, making it not a good idea to directly calculate the similarities between trajectory sequences. Thus, we instead use the location and time distribution. It will be more appropriate to transfer the trajectories and IP address distributions into vectors, and to use the cosine similarities to stand for the co-occurrence similarities of users.

To turn the location records into vectors, we use the TF-IDF model [14], which is usually used in natural language processing. We adopt the model and extend it to represent properties of location distribution. Different from [16], we build two different indexes, the spatial index and the spatial-temporal index. This is because the temporal information also have some relationship with user similarities. For example, if the trajectories of two user IDs appear at the same

```
1: function GetTfidfVector($T$)
2:     $A \leftarrow$ EmptyMap
3:     for $(x_k, y_k, t_k, w_k) \in T$ do
4:         $x_g \leftarrow \lfloor \frac{x_k}{s_g} \rfloor, y_g \leftarrow \lfloor \frac{y_k}{s_g} \rfloor, t_g \leftarrow \lfloor \frac{t_k}{\Delta t} \rfloor$,
5:         $g_{st} \leftarrow (x_g, y_g, t_g)$
6:         $t_p \leftarrow \lfloor \frac{t_k}{s_t} \rfloor$
7:         if $A$ does not have key $t_p$ then
8:             $A[t_p] \leftarrow$ EmptyMap
9:         if $A[t_p]$ does not have key $g_{st}$ then
10:            $A[t_p][g_{st}] \leftarrow 0$
11:        $A[t_p][g_{st}] \leftarrow A[t_p][g_{st}] + w_k$
12:    $\boldsymbol{f}^s, \boldsymbol{f}^{st} \leftarrow$ indexes with zero entries
13:    for $t_p$ in the key set of $A$ do
14:        $n_{t_p} \leftarrow$ number of distinct records in $t_p$
15:        for $g_{st}$ in the key set of $A[t_p]$ do
16:            $g_s \leftarrow$ the spatial index of $g_{st}$
17:            $\boldsymbol{f}^s_{g_s} \leftarrow \boldsymbol{f}^s_{g_s} + \frac{A[t_p][g]}{n_{t_p}}$
18:            $\boldsymbol{f}^{st}_{g_{st}} \leftarrow \boldsymbol{f}^{st}_{g_{st}} + \frac{A[t_p][g]}{n_{t_p}}$
19:    for each spatial grid $g_s$ do
20:        tf$_{g_s} \leftarrow \log(1 + \boldsymbol{f}^s_{g_s})$
21:        $N_{g_s} \leftarrow$ number of users appearing in $g_s$
22:        idf$_{g_s} \leftarrow \frac{1}{\log(1+N_{g_s})}$
23:        $\boldsymbol{v}^s_{g_s} =$ tf$_{g_s} \cdot$ idf$_{g_s}$
24:    for each spatial-temporal grid $g_{st}$ do
25:        tf$_{g_{st}} \leftarrow \log(1 + \boldsymbol{f}^{st}_{g_{st}})$
26:        $N_{g_{st}} \leftarrow$ number of users appearing in $g_{st}$
27:        idf$_{g_{st}} \leftarrow \frac{1}{\log(1+N_{g_{st}})}$
28:        $\boldsymbol{v}^{st}_{g_{st}} =$ tf$_{g_{st}} \cdot$ idf$_{g_{st}}$
29:    return $(\frac{\boldsymbol{v}^s}{\|\boldsymbol{v}^s\|}, \frac{\boldsymbol{v}^{st}}{\|\boldsymbol{v}^{st}\|})$
```

Fig. 5: TF-IDF vector generating

place in the same day, then the similarity of the two user IDs should be stronger than the similarity when they appear at the same place but in different days. To take full advantage with this time-based information, we also propose spatial-temporal indexes in this paper.

In our dataset, we split the whole city into small grids. We map each spatial point (x, y) into a grid $(\lfloor \frac{x}{s_g} \rfloor, \lfloor \frac{y}{s_g} \rfloor)$, and map each spatial-temporal point (x, y, t) into a grid $(\lfloor \frac{x}{s_g} \rfloor, \lfloor \frac{y}{s_g} \rfloor, \lfloor \frac{t}{\Delta t} \rfloor)$, where s_g is the size of the grids and $\Delta t = 1\,\text{day}$ is the length of time bins. Following this approach, each grid g_j (either a spatial-temporal grid or a spatial grid) can be considered as a word, and each trajectory T_i can be considered as a document made up of all grids (words) that a user has visited. The appearance of a user in the grid g_j can be regarded as the appearance of the word in the document. Similar to the TF-IDF model in document processing [14], we define the following notions.

- The term frequency (TF) of grid g_j in trajectory T_i is:

$$\text{tf}(g_j, T_i) = \log(1 + f_{i,j}),$$

  where $f_{i,j}$ is the frequency that g_j appears in T_i.
- The inverse document frequency (IDF) of grid g_j is:

$$\text{idf}(g_j) = \frac{1}{\log(1 + |\{i | f_{i,j} > 0\}|)}.$$

Then, define the TF-IDF value for grid g_j in trajectory T_i:

$$\text{tf-idf}(g_j, T_i) = \text{tf}(g_j, T_i) \cdot \text{idf}(g_j)$$

Then, given trajectory T_i, we can represent it as a vector:

$$\boldsymbol{v}(T_i) = (\text{tf-idf}(g_1, T_i), \text{tf-idf}(g_2, T_i), \cdots, \text{tf-idf}(g_{|G|}, T_i)).$$

In practice, we can instead use the normalized vector: $\boldsymbol{v}^*(T_i) = \frac{\boldsymbol{v}(T_i)}{\|\boldsymbol{v}(T_i)\|}$.

As we can see, if we want to calculate the vector for a trajectory T_i, the only thing we need to do it to estimate its appearance frequency $f_{i,j}$ for all j, and then use the above definition to calculate the vector $\boldsymbol{v}^*(T_i)$. In fact, for each trajectory T_i, we get two vectors in the same way, a spatial vector $\boldsymbol{v}^{s*}(T_i)$ and a spatial-temporal vector $\boldsymbol{v}^{st*}(T_i)$.

The frequencies $f_{i,j}$ are calculated as follows. First, split the whole time range into several 1-hour periods, and each time t is in a time period with ID $\lfloor \frac{t}{s_t} \rfloor$, where $s_t = 1\,\text{h}$. Denote the set of time periods by G_t. In a trajectory, for each record (x_k, y_k, t_k), if it is the only record in the period $\lfloor \frac{t_i}{s_t} \rfloor$, it makes a contribution of 1 to the frequency of the grid $(\lfloor \frac{x_k}{s_g} \rfloor, \lfloor \frac{y_k}{s_g} \rfloor)$ and $(\lfloor \frac{x_k}{s_g} \rfloor, \lfloor \frac{y_k}{s_g} \rfloor, \lfloor \frac{t_k}{\Delta t} \rfloor)$. If there are n_p records in a period, each record makes an equivalent contribution of $\frac{1}{n_p}$ to its grid. Summing these contributions, the frequencies $f_{i,j}$ for each grid j can be attained. For a predicted IP location distribution with a confidence value (see Section 4), i.e., (x_k, y_k, t_k, w_k), where w_k is the confidence value of the prediction, it makes a contribution of w_k for the single appearance in a period, and a contribution of $\frac{w_k}{n_p}$ for each record for the n_p multiple appearances in a period.

The complete TF-IDF generating algorithm is in Fig. 5. Here we represent the exact location records (x_k, y_k, t_k) as location predictions with a confidence of 1, i.e., $(x_k, y_k, t_k, 1)$. After we compute the vectors for each user ID in multiple spaces, we use it to compute the pairwise similarities of users in the next step.

5.3 Computing with the Pairwise Index

In Section 5.2, we have proposed a way to transform trajectories and search query records into a vector space. In order to match similar distributions in datasets, we need to compute the similarity between each pair of vectors from the two datasets, and keep the ID pairs with high similarities.

This is in fact an all-pair similarity search problem, which has been discussed, e.g., in [27], [28]. However, these algorithms require a huge amount of calculation on our datasets, which makes them incompatible to be used directly for our problem. For example, in [28], the vectors are transferred into inverted indexes, generating pairs of vectors in each index, and merging the same pairs to get similarity values. The step of enumerating all pairs of vectors with the same index takes an $O(n^2)$ complexity. To resolve this problem that existing methods are not applicable to our large dataset, we have made an improvement by building a pairwise index instead of indexes with single entries.

Next, we define the all-pair similarity problem formally. Given the two sets of vectors generated in the last step, we have the following.

- For the dataset of user trajectories, we have $\mathcal{D} = \{(v^s_1, v^{st}_1), (v^s_2, v^{st}_2), \cdots, (v^s_{|\mathcal{D}|}, v^{st}_{|\mathcal{D}|})\}$, where v^s_k and

v_k^{st} are nonnegative spatial and spatial-temporal vectors representing the location and time distribution for device ID id_k^{L}.

- For the dataset of mobile search queries, we have $\mathcal{K} = \{(u_1^s, u_1^{st}), (u_2^s, u_2^{st}), \cdots, (u_{|\mathcal{K}|}^s, u_{|\mathcal{K}|}^{st})\}$, where u_j^s and u_j^{st} are the nonnegative spatial and spatial-temporal vector representing for the location and time distribution for cookie ID id_j^{Q}.

The goal of this step is that, for each cookie ID id_j^{Q}, we want to find K different devices IDs such that they have the top-K largest spatial cosine similarities with u_j^s, and compute the spatial and spatial-temporal similarities of them with u_j^s and u_j^{st}.

Note that v_k^s, v_k^{st} and u_j^s, u_j^{st} are usually sparse vectors with only a few positive entries. We only store the positive entries during computation. We represent v_k^s and v_k^{st} as:

$$v_k^s = (v_{k,1}^s, v_{k,2}^s, \cdots) = \{\langle g_{i_1}^s, v_{k,i_1}^s\rangle, \langle g_{i_2}^s, v_{k,i_2}^s\rangle, \cdots\},$$

$$v_k^{st} = (v_{k,1}^{st}, v_{k,2}^{st}, \cdots) = \{\langle g_{h_1}^{st}, v_{k,h_1}^{st}\rangle, \langle g_{h_2}^{st}, v_{k,h_2}^{st}\rangle, \cdots\},$$

where $v_{k,i_1}^s, v_{k,i_2}^s, \cdots$ are the nonzero entries in the trajectory, and $g_{i_1}^s, g_{i_2}^s, \cdots$ are the grid IDs of these entries, and the same is with v_k^{st}. Similarly, we represent u_j^s and u_j^{st} as

$$u_j^s = (u_{j,1}^s, u_{j,2}^s, \cdots) = \{\langle g_{i_1}^s, u_{j,i_1}^s\rangle, \langle g_{i_2}^s, u_{j,i_2}^s\rangle, \cdots\},$$

$$u_j^{st} = (u_{j,1}^{st}, u_{j,2}^{st}, \cdots) = \{\langle g_{h_1}^{st}, u_{j,h_1}^{st}\rangle, \langle g_{h_2}^{st}, u_{j,h_2}^{st}\rangle, \cdots\}.$$

Our improved algorithm is theoretically based on the uniqueness bound of human mobility, which was discussed in [12], [13]. According to [12], [13], taken several points from a person's trajectory, the probability that there exists a unique trajectory which contains these points increases with the number of selected points. Based on an analysis on our dataset, the probability that a single point can be used to identify a unique person is less than 10%. This means that if two user IDs only have co-occurrences in a single grid, it will be unlikely that they can be concluded to belong to the same person. Furthermore, according to the analysis on our dataset, for the user pairs in the ground truth, the probability that two users have co-occurrences in a single grid is 16.3% in Beijing, and 20.5% in Harbin, which is not the majority in the ground truth dataset. Thus, we will only consider the case that the two user IDs have co-occurrences in at least two grids.

In the following, we only describe how to compute the spatial similarities, and the spatial-temporal similarities can be computed in almost the same way. The only difference is that we will keep all the similarities in the spatial-temporal computing, instead of only selecting the top-K similarities.

We build an index in which each key is a pair of grid cells instead of a single grid cell. In this way, we can only consider pairs of IDs who have appeared in at least two different grid cells, and the total time and space requirement can be reduced significantly in this way. The main idea of the pairwise index is shown in Fig. 6.

Since the dataset is large within a city, our algorithm runs in parallel in a Hadoop Map-Reduce framework. Our algorithm consists of two steps. In the first step, we build an inverted pairwise index from the vectors for the users. In the mapper stage, for each pair of nonzero entries $\langle g_{i_p}^s, v_{k,i_p}^s\rangle, \langle g_{i_q}^s, v_{k,i_q}^s\rangle$ in vector v_k^s, we generate a key-value pair, where the key is the grid pairs $(g_{i_p}^s, g_{i_q}^s)$, and the value is made up of the user ID id_k^{L} and the entry weights $(v_{k,i_p}^s, v_{k,i_q}^s)$. In the reducer stage, we merge all values $\langle id_k^{\mathrm{L}}, (v_{k,i_p}^s, v_{k,i_q}^s)\rangle$ into a set belonging to the same key (i.e., pairs of grids). The pseudo-code of this step is in Fig. 7.

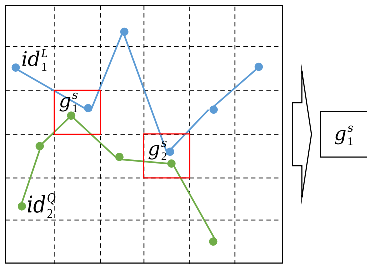


Fig. 6: Pairwise index: building index for the pairwise co-occurrences for the user ID pairs. User id_1^{L} and id_2^{Q} both appear in grid g_1^s and g_2^s. Build a pairwise index with the key of (g_1^s, g_2^s). Add id_1^{L} and id_2^{Q} into the ID list of this index.

```
1: function MAPPER1($(\langle id_k^{\mathrm{L}}, v_k^s\rangle)$)
2:     for $\langle g_{i_p}^s, v_{k,i_p}^s\rangle, \langle g_{i_q}^s, v_{k,i_q}^s\rangle \in v_k^s$ do
3:         output $(g_{i_p}^s, g_{i_q}^s)$, $\langle id_k^{\mathrm{L}}, (v_{k,i_p}^s, v_{k,i_q}^s)\rangle$
4: function MAPPER2($(\langle id_j^{\mathrm{Q}}, u_j^s\rangle)$)
5:     for $\langle g_{i_p^s}, u_{j,i_p}^s\rangle, \langle g_{i_q}^s, u_{j,i_q}^s\rangle \in u_j^s$ do
6:         output $(g_{i_p}^s, g_{i_q}^s)$, $\langle id_j^{\mathrm{Q}}, (u_{j,i_p}^s, u_{j,i_q}^s)\rangle$
7: function REDUCER($((g_{i_p}^s, g_{i_q}^s), set(\langle id_k^{\mathrm{L}}, (v_{k,i_p}^s, v_{k,i_q}^s)\rangle),$
                      $set(\langle id_j^{\mathrm{Q}}, (u_{j,i_p}^s, u_{j,i_q}^s)\rangle))$)
8:     output $\langle (g_{i_p}^s, g_{i_q}^s), set(\langle id_k^{\mathrm{L}}, (v_{k,i_p}^s, v_{k,i_q}^s)\rangle),$
                      $set(\langle id_j^{\mathrm{Q}}, (u_{j,i_p}^s, u_{j,i_q}^s)\rangle)\rangle$
```

Fig. 7: Step One of similarity computation: generating the pairwise indexes

In the second step, we compute the similarity between pairs of IDs with at least two co-occurrences. For each cookie ID, we keep the K device IDs with the largest similarities with the cookie ID. Note that we have already obtained an inverted list in Step 1, i.e.,

$$\langle (g_{i_p}^s, g_{i_q}^s), set(\langle id_k^{\mathrm{L}}, (v_{k,i_p}^s, v_{k,i_q}^s)\rangle), set(\langle id_j^{\mathrm{Q}}, (u_{j,i_p}^s, u_{j,i_q}^s)\rangle)\rangle.$$

In the mapper stage, for each pair of grids $(g_{i_p}^s, g_{i_q}^s)$, we match each item $\langle id_k^{\mathrm{L}}, (v_{k,i_p}^s, v_{k,i_q}^s)\rangle$ from the device ID set and item $\langle id_j^{\mathrm{Q}}, (u_{j,i_p}^s, u_{j,i_q}^s)\rangle$ from the cookie ID set. The first key of the output is id_j^{Q}, the second key is id_k^{L}, and the value is made up of the grid IDs and the product of the vector entries, i.e., $\langle g_{i_p}^s, v_{k,i_p}^s u_{j,i_p}^s\rangle$ and $\langle g_{i_q}^s, v_{k,i_q}^s u_{j,i_q}^s\rangle$. In the reducer stage, we merge the items with the same pair of user IDs $(id_j^{\mathrm{Q}}, id_k^{\mathrm{L}})$, and sum all products of vector entries, i.e., $v_{k,i_p}^s u_{j,i_p}^s$ for all p, to obtain similarities between these two user IDs, i.e. $\sum_p v_{k,i_p}^s u_{j,i_p}^s$. Finally, for each cookie ID id_j^{Q}, we process a sorting to all device IDs by similarities with id_j^{Q}, and keep the top-K most similar IDs. The pseudo-code of this step is in Fig. 8. In this way, we compute the best matched device IDs for each cookie ID together with similarities.

```
1: function MAPPER($\langle(g^s_{i,p}, g^s_{i,q}), set(\langle id^{\mathrm{L}}_k, (v^s_{k,i_p}, v^s_{k,i_q})\rangle), set(\langle id^{\mathrm{Q}}_j, (u^s_{j,i_p}, u^s_{j,i_q})\rangle)\rangle$)
2:     for $\langle id^{\mathrm{L}}_k, (v^s_{k,i_p}, v^s_{k,i_q})\rangle$ do
3:         for $\langle id^{\mathrm{Q}}_j, (u^s_{j,i_p}, u^s_{j,i_q})\rangle$ do
4:             output $id^{\mathrm{Q}}_j, id^{\mathrm{L}}_k, \langle g^s_{i_p}, v^s_{k,i_p}u^s_{j,i_p}\rangle$
5:             output $id^{\mathrm{Q}}_j, id^{\mathrm{L}}_k, \langle g^s_{i_q}, v^s_{k,i_q}u^s_{j,i_q}\rangle$
6: function REDUCER($id^{\mathrm{Q}}_j, set(\langle id^{\mathrm{L}}_k, set(\langle g^s_{i_p}, v^s_{k,i_p}u^s_{j,i_p}\rangle)\rangle)$)
7:     $sim \leftarrow$ EMPTYMAP
8:     for $\langle id^{\mathrm{L}}_k, set(\langle g^s_{i_p}, v^s_{k,i_p}u^s_{j,i_p}\rangle)\rangle$ do
9:         $sum \leftarrow 0$
10:        for $g^s_{i_p}, v^s_{k,i_p}u^s_{j,i_p}$ do
11:            $sum \leftarrow sum + v^s_{k,i_p}u^s_{j,i_p}$
12:        $sim[id^{\mathrm{L}}_k] = \langle\langle id^{\mathrm{Q}}_j, id^{\mathrm{L}}_k\rangle, sum\rangle$
13:    output top-$K$ from $sim$
```

Fig. 8: Step Two of similarity computation: getting the top-K matched ID pairs

5.4 Index with Time Drifts

In section 5.2, we define the spatial-temporal index which could be used to compute the co-occurrences of user trajectories. However, in the temporal dimension of the index, we have split the whole time period into small time intervals of $\Delta t = 1\,\mathrm{day}$, and in this way, there may be pairs of nodes closed to each other, but not laying in the same time interval. To tackle this problem, we introduce a novel variant of the index, which is named as index with time drifts.

In the new index, we use $\Delta t' = \frac{1}{2}\Delta t$ as the new time interval length. In the algorithm of Fig. 5, for each node (x_k, y_k, t_k), its spatial-temporal index turns to be:

$$(x_g, y_g, t'_g) = (\lfloor\frac{x_k}{s_g}\rfloor, \lfloor\frac{y_k}{s_g}\rfloor, \lfloor\frac{t_k}{\Delta t'}\rfloor)$$

For each node in the offline dataset, we directly use (x_g, y_g, t'_g) as its index. For the online dataset, we add the drifts of $\{-1, 0, 1\}$ to each time index, i.e., for each node (x_k, y_k, t_k), we build three time indexes for it: (x_g, y_g, t'_g-1), (x_g, y_g, t'_g), (x_g, y_g, t'_g+1). We repeat line 9-11 of Fig. 5 for three times, and in each repetition, use each of the above three indexes to replace g_{st}. Other parts of the algorithm remain the same. By this variation of index with drifts, we can benefit in these ways: (a) All the co-occurrences in the Δt intervals can also be included in the $\Delta t'$ intervals with drifts. (b) No co-occurrence will be ignored as long as the time difference is no more than $\Delta t'$. In this way, two pieces of trajectory nodes will not be assigned into different time slots if their occurrence time is quite close. Therefore, we can attain a higher coverage of user co-occurrences.

The index with time drifts has better effect if the users's behavior does not have clear inactive time boundary. In our experiment, we divide the time bins at the midnight times, and these are usually inactive time for users. If there is a such clear inactive time for most of users, different ID's nodes of the same user should have little chance to be divided into different time intervals. However, in a big city like Beijing, such global inactive time boundary may not be clear since the city's nightlife is rich and varied. Whereas, in a middle city like Harbin (a provincial capital city in Northeast China), most of users are not active at the midnight. Our experiment in Section 7.3.2 also demonstrates such interesting phenomenon that the index with time drifts has improved performance on the dataset of Beijing, but has little effect on the dataset of Harbin.

5.5 Optimization of Pairwise Indexes

In Section 5.3, we have described an algorithm for building pairwise indexes. In this section, we propose how to optimize the pairwise index for the spatial similarity calculation.

At first, in order to reduce the size of the intermediate data, for each inverted list we split the device ID set and the cookie ID set into several subsets, say $\{S_{\mathrm{L},1}, \cdots, S_{\mathrm{L},N_{\mathrm{L}}}\}$ and $\{S_{\mathrm{Q},1}, \cdots, S_{\mathrm{Q},N_{\mathrm{Q}}}\}$. For each iteration, we take one subset $S_{\mathrm{L},i}$ from the device ID set and one subset $S_{\mathrm{Q},j}$ from the cookie ID set, computing the top-K similar device IDs for each cookie ID. After matching up all pairs of these subsets, we merge these lists together to get the top-K similar device IDs for each cookie ID in the whole set.

However, as we will explain, not all pairwise grids need to be used in building inverted lists. For example, for a pair of users, if they are in pair-indexes (g_1, g_2) and (g_1, g_3), it is clear that they have co-occurrences at grids g_1, g_2 and g_3. Then, there is no need to build the pair-index (g_2, g_3) any more. However, the problem is that we do not know which pairs are not needed. Nevertheless, if we have a threshold for the minimum value of similarity, it will be possible to reduce the size of the index. Since we have split the user sets into several subsets, the threshold can be attained by the result of the last subset. For example, if we have computed the top-K similarities between cookie ID subset $S_{\mathrm{Q},1}$ and device ID subset $S_{\mathrm{L},1}$, then when computing the similarity of $S_{\mathrm{Q},1}$ and $S_{\mathrm{L},2}$, for each id^{Q}_k in $S_{\mathrm{Q},1}$, we can assign its threshold r_k in $S_{\mathrm{L},2}$ to be the K-th largest similarity value for id^{Q}_k among $S_{\mathrm{L},1}$. When a user in $S_{\mathrm{L},2}$ has a similarity no more than r_k with id^{Q}_k, we can just ignore it.

The problem becomes the following. For each device ID id^{L}_k and a threshold r_k, our goal is to find all users $u^s_j \in \mathcal{K}$, such that the inner product $\langle v^s_k, u^s_j\rangle > r_k$, and then compute the similarity values $s_{k,j} = \langle v^s_k, u^s_j\rangle$ for all these u^s_j.

Here we solve this new problem. Let $u^s_{\max,i} = \max_j u^s_{j,i}$. Define the co-occurrence set of v^s_k and u^s_j to be the set of grids that both users appear: $C_{k,j} = \{i \mid v^s_{k,i} > 0, u^s_{j,i} > 0, 1 \le i \le M\}$, where M is the number of grids. Let $w_{k,i} = v^s_{k,i}u^s_{\max,i}$. Then, the inner product of v^s_k and u^s_j satisfies

$$\langle v^s_k, u^s_j\rangle = \sum_{i\in C_{k,j}} v^s_{k,i}u^s_{j,i} \le \sum_{i\in C_{k,j}} v^s_{k,i}u^s_{\max,i} \le \sum_{i\in C_{k,j}} w_{k,i}.$$

If $\sum_{i\in C_{k,j}} w_{k,i} \le r_k$, then $\langle v^s_k, u^s_j\rangle \le r_k$, and this means the similarity value is not large enough. Then we only need to consider the pairs (v^s_k, u^s_j) such that $\sum_{i\in C_{k,j}} w_{k,i} > r_k$.

To build a pairwise index for v^s_k, we define a graph $G_k = (V_k, E_k)$, such that $V_k = \{i \mid v^s_{k,i} > 0, 1 \le i \le M\}$ is the set of all grids v^s_k appears, and $E_k \subseteq \{(i,j) | i < j, i, j \in V_k\}$ is the key set of the pairwise index. Then for each subset $V' \subseteq V_k$, as long as V' satisfies $\sum_{i\in V'} w_{k,i} > r_k$, then it is possible that there exists a user u^s_j with $C_{k,j} = V'$, and that means $\sum_{i\in C_{k,j}} w_{k,i} > r_k$, and we should preserve that this ID pair should not be discarded. In this case, for the subgraph $G_k(V') = (V_k(V'), E_k(V'))$ where $V_k(V') = V_k \cap$

V' and $E_k(V') = \{(i,j) \in E_k \mid i,j \in V'\}$, for each node $i \in V_k(V')$, we need the degree of i in $G_k(V')$ to be at least 1, so that the entry $v^s_{k,i}$ could be indexed in at least one pair in $E_k(V')$.

With these conditions, our problem becomes the following. Given a node set V_k, assigning a weight $w_{k,i} \geq 0$ for each $i \in V_k$, we need to find an edge set $E_k \subseteq \{(i,j) | i < j, i,j \in V_k\}$, such that for each $V' \subseteq V_k$ which satisfies $\sum_{i \in V'} w_{k,i} > r_k$, the subgraph of $G_k = (V_k, E_k)$ on V' must not have a zero-degree node. The edge set E_k should be as small as possible to reduce the size of the index.

We solve this problem in such a way. We will try to split V_k into several non-intersected sets, i.e., $V_k = V_{k,1} \cup \cdots \cup V_{k,t} \cup V^*_k$, such that $\sum_{i \in V_{k,r}} w_{k,i} \leq r_k$ for each $1 \leq r \leq t$, and $w_{k,i} > r_k$ for $i \in V^*_k$. Let the edge set be

$$E_k = \left(\bigcup_{1 \leq i < j \leq t} V_{k,i} \times V_{k,j} \right) \cup \left(\bigcup_{i=1}^{t} V_{k,i} \times V^*_k \right) \cup \{(i,j) | i < j, i,j \in V^*_k\}$$

Then it is obvious that E_k satisfies our requirement. We can use all the node pairs in E_k to replace the complete index $\{(i,j) | i < j, i,j \in V_k\}$ for V_k.

Here we propose a greedy algorithm for it. At first, we sort all the nodes in V_k with nonnegative weights $w_{k,i}$ in an increasing order. Then, we initialize the first subset by $V_{k,1} = \emptyset$, and repeatedly put the smallest remaining item in V_k into this subset if the sum of the elements in $V_{k,1}$ does not exceed the threshold r_k. When the sum of the weights in $V_{k,1}$ would exceeds r_k once new elements are included, we set up a new subset $V_{k,2}$ and put the following elements into this new set. We repeat the process for $V_{k,3}, V_{k,4}, \cdots$ until all the elements with a weight no more than r_k have been put in one subset $V_{k,i}$. After that, we put the remaining elements with weights larger than r_k into the remaining subset V^*_k. Finally, for each pair of elements in the same subset $V_{k,i}$, we will not build the inverted list for them. In this way, the size of the index can be reduced.

6 Ranking with Different Features

In Section 5, we describe an algorithm to generate the top-K similar device IDs with a cookie ID by the TF-IDF similarity. However, except for the TF-IDF similarity, there are still other features that can be used to describe user similarities. After getting the top-K device IDs via TF-IDF similarities, it is useful to take other features into account rather than only using the TF-IDF similarities. For this purpose, we use the GBDT [33] to implement a learning-to-rank (LTR) model [15] to get a general ranking for all the top-K IDs related to the same cookie ID, which can be used to identify whether two IDs belong to the same user. Note that we cannot skip the pre-filtering step based on TF-IDF similarities described in Section 5, since directly applying the LTR model on all ID pairs will incur a high computation cost.

The features for the LTR we will use include: (1) The spatial TF-IDF similarity between two users described in Section 5. (2) The spatial-temporal TF-IDF similarity. (3) The number of spatial co-occurrence grids for two users. (4) The number of spatial-temporal co-occurrence grids. (5) The maximum distance between pairs of grids at which two users have co-occurrences. That is, if two users meet at p different grids $g_{i_1}, g_{i_2}, \cdots, g_{i_p}$, then this distance will be $\max_{1 \leq j < k \leq p} \mathrm{dis}(g_{i_j}, g_{i_k})$. (6) Similarities between the sets of IP addresses,[1] mobile phone operation systems and mobile phone models. These similarities are calculated using the Jaccard similarity coefficient of two sets S_1 and S_2, where S_1 and S_2 are the feature sets for the two users.

Compared with the LTR process in [16], we also add the spatial-temporal TF-IDF similarities as input feature. After merging all these related features, we can get a vector x_i for each ID pair that can be used to train the ranking model. In our experiment, we use a small set of user login records as ground truth to train the model to decide whether two IDs belong to the same user. Let $y_i = 1$ if the i-th pair of IDs have ever shared the same login account, and $y_i = 0$ if not. Our objective is to train a prediction model $f(\cdot)$, which minimizes the sum of the training losses between $\hat{y}_i = f(x_i)$ and y_i for all ID pairs on training data. In our implementation, we use the Gradient Boosting Decision Tree (GBDT) introduced in [33] to construct the model $f(\cdot)$.

While training the LTR model, we have selected 80% of all the samples as the training set, 10% as the validation set, and the remaining 10% as the testing set. After the training process, we get a ranking model $f(\cdot)$, which can return how similar a device ID is to a cookie ID. Using this ranking model, we get a prediction about which device ID should be chosen to be the best-matched result among all the top-K similar IDs.

7 Experiments

In order to evaluate the efficiency and effectiveness of our framework, we have designed several experiments.

7.1 Experiment settings

Our experiments are based on the dataset of mobile query logs and trajectories in two cities: (1) Beijing, a megacity and the capital of China, and (2) Harbin, a provincial capital city in Northeast China. Both datasets are randomly sampled as a portion of whole data. Specifically, in this experiment, our goal is to match the users IDs from the datasets of mobile query logs and trajectories, and the time periods are June 2016 for Beijing and December 2015 for Harbin. In our experiments, we only use IDs with an active period of at least a week and visited at least two different places (or at least two IP addresses). Our datasets are as follows.

- Mobile query log dataset: Obtained from the search query logs of the Baidu search box generated from the mobile phones. 40.7% of all the query records have location information, while the rest 59.3% only have raw IP addresses.
- Trajectory dataset: Obtained from the location records from a mobile application "Mobile Baidu".[2]

Table 1 shows the statistics of our datasets, including the number of IDs and records, and the average number of records for each ID for each dataset.

1. The trajectory data may also contain IP addresses for uploading the location information.
2. http://xbox.m.baidu.com/wuxian/

TABLE 1: Statistics of the datasets of two cities

City		Beijing	Harbin
Query Logs	# of IDs	$12,616,442$	$2,410,892$
	# of records	$1,829,633,176$	$299,708,763$
	Records per ID	145	124
Trajectories	# of IDs	$2,811,169$	$748,789$
	# of records	$574,008,583$	$142,768,995$
	Records per ID	204	191

Our framework is implemented as a series of Hadoop streaming jobs written in Python 2.7, and the experiments have been running on the Hadoop Map-Reduce [34] cluster with $3,000$ nodes running in parallel for each job. All the reported results are the averages of five running results.

Our default parameters are $s_g = 30\,\mathrm{m}$, $K = 100$ and $\Delta t = 24\,\mathrm{h}$. For the optimization in Section 5.5, we have $N_{\mathrm{Q}} = N_{\mathrm{L}} = 2$. Since performance of the index with time drifts method depends on the data property (which has improvement in the dataset of Beijing, but not improvement in the dataset of Harbin), therefore we disable the index with time drifts method by default. In the learning-to-rank step, we set the step size to be $\eta = 0.1$, minimum loss reduction to be $\gamma = 2$, minimum child weight to be 0.05, and number of rounds to be 50. For the decision tree depth, we set it to $d_t = 9$ to get the best validation result.

The ground truth comes from records of account logins. If a device ID has been logged as the same user account, as a cookie ID, then we can use this device-cookie ID pair as a ground truth. In our dataset, there are $87,456$ cookie IDs in Beijing and $21,518$ in Harbin which have related device IDs, and we use these ID pairs as ground truth.

7.2 Running time and space cost

As discussed before, since this is a problem in which pairs of user IDs are needed to be matched to compute the similarity, the time and space complexity is $O(n^2)$ with respect to the size of the user ID set. Here we have designed a series of experiments in order to test the efficiency of our framework.

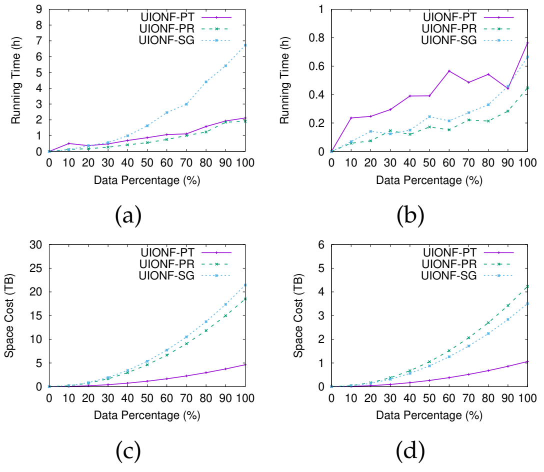


Fig. 9: Total run times for Beijing (a) and Harbin (b), and total space costs for Beijing (c) and Harbin (d) for different indexes

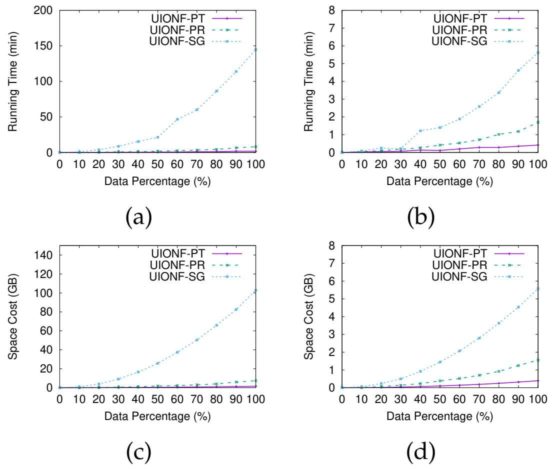


Fig. 10: Maximum single mapper run times for Beijing (a) and Harbin (b), and maximum single mapper space costs for Beijing (c) and Harbin (d) for different indexes

The total run time and space cost of our framework are plotted in Fig. 9. In each iteration, we extract both the percentage $10\%, 20\%, \cdots, 100\%$ from the set of the cookie IDs and the device IDs. In Fig. 10, we plot the maximum values of the run times and space costs for all the mappers in the Map-Reduce jobs. Note that the main time and space cost of our framework is due to the second step to compute the TF-IDF similarities described in Section 5. In each iteration, we test three different methods on the subset extracted, and get the time costs and maximum memory usages during the whole processes of computing TF-IDF similarities. The main differences of the compared methods are how to build the index for computing similarity, including:

- UIONF-PT: the pairwise index in Section 5.
- UIONF-PR: the pairwise index without the optimization method described in Section 5.5.
- UIONF-SG: using vanilla inverted index in [16] instead of the proposed pairwise index.

From Fig. 9(a)(b), we see that when the dataset is small (e.g., for Harbin), UIONF-PT (which is with pairwise index with optimization) takes much more time, because when splitting the whole problem into small sub-jobs, it takes more time for the initialization of the jobs. When the dataset becomes larger, the gap is not as large. Particularly, it takes much time for single index when the dataset gets larger because the problem of extreme length of some posting lists leads to imbalance problem, and these huge lists cause instability to the Hadoop cluster such that some subtasks will keep crashing and restarting, which takes a lot of time.

From Fig. 9 (b), we can see that the maximum space requirement grows quadratically with the size of the dataset. It can be seen that less space will be used when we use UNICORN-PT (which adopts pairwise index with optimization) than the other two methods. In our experiment, UIONF-PR takes less space than single index in Beijing, and a little more in Harbin, and it depends on the datasets.

As in Fig. 10, the original UIONF-SG suffers from serious imbalance issues for the mappers, and solutions with the pairwise index such as UIONF-PT and UIONF-PR can efficiently deal with this problem. For both the space and time complexities, the pairwise index with the proposed optimization method (UIONF-PT) is more efficient for a large dataset.

7.3 Effectiveness of our framework

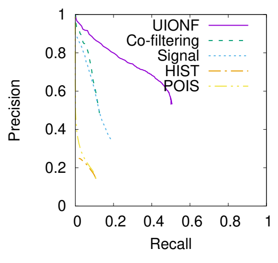
In this part, we evaluate the effectiveness of our framework based on comparison with existing methods. As introduced in Section 6, in the LTR step, we set up a filtering rule for prediction as follows. If this prediction value does not exceed some particular threshold, we consider that no device ID is matched with this cookie ID. Otherwise the device ID with the highest probability value is matched with this cookie ID. By setting up different values for the threshold, we can get a precision-recall curve for the prediction.

7.3.1 Comparison with the state-of-the-art method

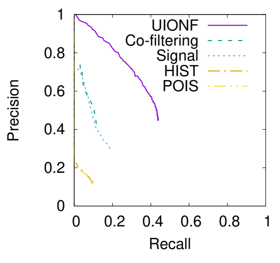
Here we compare our method with some state-of-the-art methods such as [9], [20], [21], [35]. Since these methods are designed for offline data, for the online data in our dataset, we can only use the records with exact location information. As introduced in Section 6, in our method, we use a learning-to-rank approach, using 80% of the ID pairs in the ground truth as the training set, 10% as the validation set and the other 10% as the testing set. Then we mix the 10% test ID pairs with all the device IDs to test the performance of our model. Since our task is to find the best-matched device ID from the candidate device ID set for each cookie ID, we always use the whole set of device IDs, regardless of whether we are in the training and testing process. Since there are no machine learning algorithms such as LTR in the baselines, which means there is no need for training and validation, we only use the testing ID pairs when evaluating the baselines.

We have compared our framework with the state-of-the-art location-based user identification methods in [9] and used two different similarity measures in [9]: 1) co-filtering with signal and Jaccard similarities (Co-filtering), and 2) only the signal similarities (Signal). The precision-recall curves of these two method together with our framework is shown in Fig. 11(a)(b), from which we can see that our framework get a better precision and recall on our dataset. We have also tried the baselines together with the method of IP-location clustering in Section 4, which is shown in Fig. 11(c)(d). There are two main advantages in our framework compared with existing methods: a) We have use the IP clustering approach to enrich the online query logs without location information; and b) We have used more information in the learning-to-rank to get a more accurate prediction.

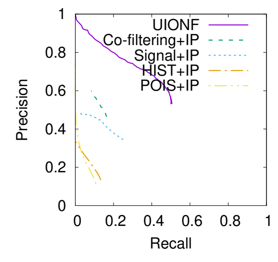
We have also compared our method with some of the probability model based methods such as HIST [20], [35] and POIS [21] which is shown in Fig. 11(a)(b). The implementation of HIST is mainly based on paper [20] since [20] is an extension of [35]. When running on a Hadoop platform, both HIST and POIS need a step of computing all the co-appearances between users with an inverted index, so they have a similar time and space complexity with UIONF-SG. However, these models do not show a good performance in our dataset. There are several reasons. (a) Similar to [9], there are not methods to fuse the online and offline data



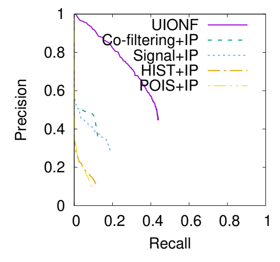
(a)



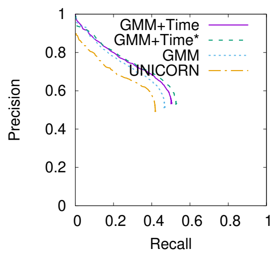
(b)



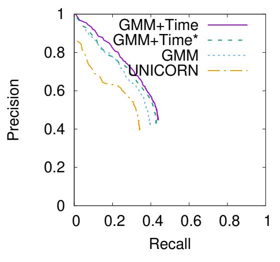
(c)



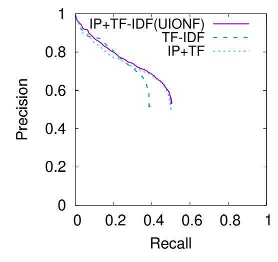
(d)



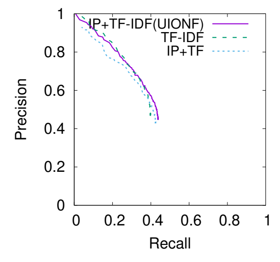
(e)



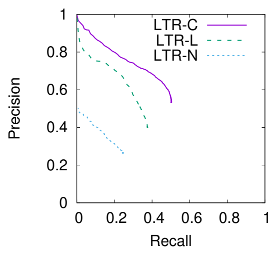
(f)



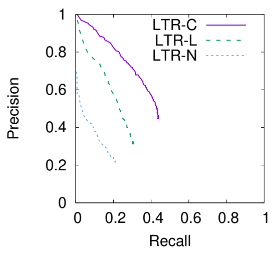
(g)



(h)

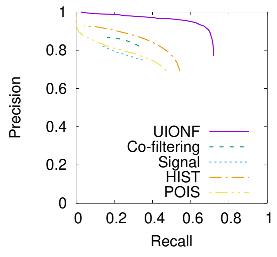


(i)

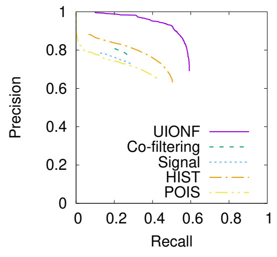


(j)

Fig. 11: Comparing our framework with baselines in (a) for Beijing and (b) for Harbin, where raw location information is used in the baselines. Comparing our framework with baselines in (c) for Beijing and (d) for Harbin, where the IP-location clustering in Section 4 is used in the baselines. Examining the performance of the GMM clustering method and spatial-temporal-based indexes in (e) for Beijing and (f) for Harbin. Examining how IP clustering and TF-IDF vectors contributed to the performance of our framework in (g) for Beijing and (h) for Harbin. Comparing the effect of different features of LTR in (i) for Beijing and (j) for Harbin.



(a)



(b)

Fig. 12: Comparing our framework with baselines in (a) for Beijing and (b) for Harbin, using only the subset of the whole dataset which appears in the ground truth.

like IP clustering and learning-to-rank methods to improve the performance of the models. (b) The method in [20], [35] assumes all the available information about a user is in the form of histograms. Some useful information for user identification cannot be contained by such histogram data. (c) It has been assumed in [21] that user trajectories follow certain probability models such as the Poisson distribution. However, user trajectories have some more complex patterns [9], which cannot be described well by these models.

In our original training process, we use the whole user dataset to compute user similarities, and evaluate the performance of our model and [9], [20], [21], [35] with the whole dataset containing all user IDs. However, most user IDs in the whole dataset do not appear in the ground truth. Now we only compute user similarities between the user IDs which appear in the ground truth, and evaluate the prediction performance on the 10% testing data of the ground truth, where we do not include all the candidate device IDs. For each cookie ID in the test set, we use our framework and the baselines to find the best-matched device ID among the candidate set of device IDs which appear in the ground truth. Since the ground truth dataset is much smaller than the whole dataset, we expect higher precision and recall values when the candidate device ID set is restricted to the ground truth set. The result on the ground truth set is shown in Fig. 12. After replacing it with the ground truth dataset, precision and recall values are higher with all methods, and our framework still has a significant improvement compared to the baselines.

For the baseline of [9], we are using the grid sizes of $\{20\,\mathrm{m}, 200\,\mathrm{m}\}$, $r_{20} = 0.625$, $r_{200} = 0.375$, $m_c = 2000$, $\eta = 16$, $\gamma = 0.2$, $\alpha = 0.4$, $\lambda = 50$, $\mu = 1/4000$, $\beta = \{0.8, 0.2\}$. For other baselines, we have chosen the grid index size of $30\,\mathrm{m}$ in [20], [35], and $(10\,\mathrm{m}, 31\,\mathrm{days})$ in [21]. In this way, the best matching results can be attained.

7.3.2 Effect of GMM clustering and spatial-temporal index

Fig. 11(e)(f) shows how GMM clustering and the spatial-temporal index contribute to our framework. "GMM+Time" is for the UNIOF framework in this work, "GMM" is for the method with GMM but without spatial-temporal index, and "UNICORN" is for the method in [16] without GMM or spatial-temporal index. From the curves, we see that after replacing DBSCAN with GMM, there is an obvious improvement with the precision-recall curves. After adding the spatial-temporal index, the effect raises even more.

We have also evaluated the index with time drifts in Section 5.4, which is referred as "GMM+Time*". This method has reasonable improvement of the performance in the dataset of Beijing, but not the one of Harbin. As we discussed in Section 5.4, the major reason is because the users in Harbin has a global inactive time for clearly splitting the time intervals. Whereas the users in Beijing do not have a global inactive time boundary due to the richful nightlife, and it hard to divide the temporal dimension without distributing temporally closed nodes into different time bins. Therefore, the index with time drifts method can bring more benefit for the dataset of Beijing. Besides, if two user IDs belong to the same user, it is usual that they have a lot of co-occurrences. It does not have much influence if only a very small part of co-occurrences are ignored.

7.3.3 Effect of IP-to-location mapping and TF-IDF vectors

We show how IP clustering contributed to our prediction in Fig. 11(g)(h). We can see that when we need a high predicting accuracy, it does not make much difference with the curves of the method with IP clustering (IP+TF-IDF) and without IP clustering (TF-IDF). If we need a high recall, IP clustering makes an obvious contribution. This is because we can take advantage of more information with IP clustering, but there are some limitations because many IP addresses distribute in a large area and it can not preserve a high accuracy.

We also show that the TF-IDF vectors (IP+TF-IDF) have a contribution to our performance compared to pure TF vectors (IP+TF). By TF-IDF, when the recall is 0.2, we can increase the precision by 2.9 percent in the dataset of Beijing, by 5.0 percent in the dataset of Harbin; and when the recall is 0.4, we can increase the precision by 1.0 percent in the dataset of Beijing, by 4.4 percent in the dataset of Harbin. TF-IDF is better because the co-occurrences in more popular grids will make a less contribution to the similarities of users. Since the computation cost of TF-IDF is small, it is reasonable to adopt the TF-IDF method instead of pure TF.

7.3.4 Comparison of different LTR methods

The precision-recall curves of our framework with different LTR methods are plotted in Fig. 11(g)(h), including:

- LTR-Complete (LTR-C): using all location-based features, together with other features such as IP addresses, mobile phone OSes and phone models.
- LTR-Location (LTR-L): using only location-based features in LTR.
- LTR-None (LTR-N): not using LTR, using only the spatial-based similarities.

From Fig. 11(i)(j), we can see that the location-based LTR (LTR-L) contributes to the precision and recall of the identification, and after adding additional information (LTR-C), we gets a better prediction result than LTR-L. When increasing the threshold, the recall decreases slightly but the precision increases significantly. One can select an appropriate threshold to adjust the precision and recall based on actual needs.

7.4 Parameter Evaluation

In this section, we evaluate the effect of parameters of our framework. At first, we evaluate the coverage of the top-K TF-IDF-based similar IDs for different values of K, as introduced in Section 5. The coverage means the percentage of real cookie ID and device ID pairs that can be retrieved in the returned top-K results. For example, with 100 real ID pairs, if there are 60 cookie IDs whose top-K device IDs contain its corresponding real device ID, we say the coverage is 60%. It is clear that the coverage define the upper bound of the recall.

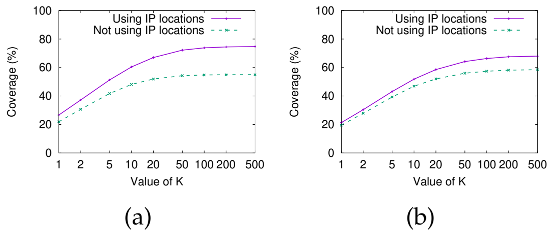


Fig. 13: Top-K coverage by TF-IDF similarities for Beijing (a) and Harbin (b).

The results of the top-K coverage have been plotted in Fig. 13. From the figure, we see that the coverage for the method with IP clustering is significant larger than that without IP clustering, which demonstrates the effectiveness of our proposed IP localization.

From Fig. 13, we see that the top-K coverage of the TF-IDF similarities increases when we rise the value of K, and it gets very close to the upper bound when the value of K is over 100. Although a better coverage can be attained when we assign a larger value of K, doing so also increases the computation cost of the following steps. Thus, we decide to assign $K = 100$ to achieve a good tradeoff. Under this value, most real ID pairs will be covered. We have also plotted the coverage curves without IP clustering, and we see that the method IP clustering increases the coverage rates.

We also evaluate how the K value and the grid size affect the prediction performance. The precision-recall relation as a function of K are illustrated in Fig. 14(a)(b). We see that when K is large enough, the precision and recall values are not very sensitive to K, which coincides with Fig. 13.

Similarly, results for different grid sizes are plotted in Fig. 14(c)(d). For the grid size, we see that the precision with the same recall will become very different when we choose different grid sizes. When the grid size is too small, most co-occurrences cannot be discovered; and when it is too large, there will be a lot of false-positive co-occurrences. From the results, we choose $s_g = 30\,\mathrm{m}$ to best fit our dataset.

For comparison purposes, we have also evaluated the decision tree depth of the GBDT approach in Fig. 14(e)(f). From the results, we see that for small depths, the precision increases with the depth, but when the depth grows large, there is no significant increment of precision.

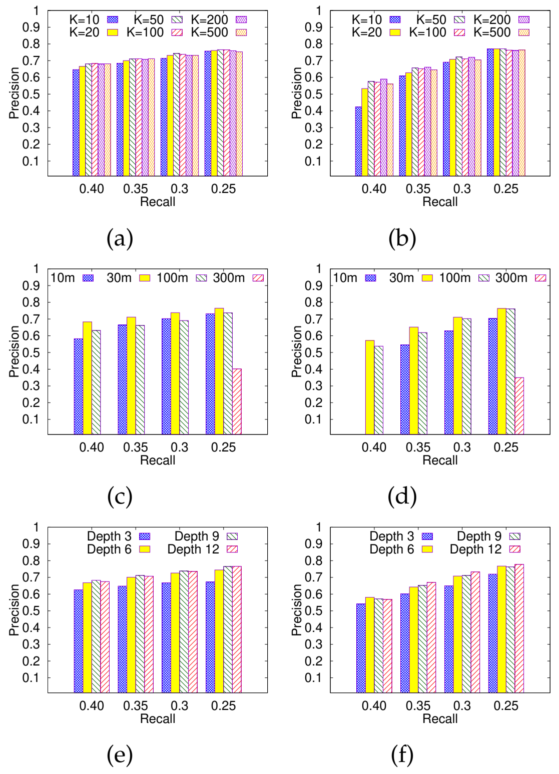


Fig. 14: Comparison of different parameters. Values of K in (a) for Beijing and (b) for Harbin. Values of s_g in (c) for Beijing and (d) for Harbin. Values of GBDT depth d_t in (e) for Beijing and (f) for Harbin.

8 Conclusions

In this paper, we propose and improve the framework in [16] to deal with the user identification problem between heterogeneous datasets from the online and offline data. Firstly, for the online mobile query records without exact location information, we develop a method by the GMM-based IP location clustering to enrich them with location data. Then, we measure the co-occurrence by TF-IDF between location distributions of query records and trajectories by building the spatial and spatial-temporal pairwise index. The pairwise index is proven to be effective in reducing time and space cost of the computation, and two kinds of indexes are adopted to increase the performance of the task. After that, we use a learning-to-rank approach to figuring out the matched ID pairs among several possible similar IDs. Our experiments demonstrate the effectiveness and improvement on the task of user identification between the datasets of mobile query logs and trajectories.

References

[1] J. Vosecky, D. Hong, and V. Y. Shen, "User identification across multiple social networks," in *NDT*, 2009, pp. 360–365.
[2] X. Kong, J. Zhang, and P. S. Yu, "Inferring anchor links across multiple heterogeneous social networks," in *CIKM*, 2013, pp. 179–188.
[3] S. Tan, Z. Guan, D. Cai, X. Qin, J. Bu, and C. Chen, "Mapping users across networks by manifold alignment on hypergraph." in *AAAI*, 2014, pp. 159–165.
[4] S. Liu, S. Wang, F. Zhu, J. Zhang, and R. Krishnan, "Hydra: Large-scale social identity linkage via heterogeneous behavior modeling," in *SIGMOD*, 2014, pp. 51–62.
[5] O. Goga, P. Loiseau, R. Sommer, R. Teixeira, and K. P. Gummadi, "On the reliability of profile matching across large online social networks," in *KDD*, 2015, pp. 1799–1808.
[6] X. Zhou, X. Liang, H. Zhang, and Y. Ma, "Cross-platform identification of anonymous identical users in multiple social media networks," *TKDE*, vol. 28, no. 2, pp. 411–424, 2016.
[7] H. Zang and J. Bolot, "Anonymization of location data does not work: A large-scale measurement study," in *MobiCom*, 2011, pp. 145–156.

[8] L. Rossi, J. Walker, and M. Musolesi, "Spatio-temporal techniques for user identification by means of GPS mobility data," *EPJ Data Science*, vol. 4, no. 1, p. 1, 2015.

[9] W. Cao, Z. Wu, D. Wang, J. Li, and H. Wu, "Automatic user identification method across heterogeneous mobility data sources," in *ICDE*, 2016, pp. 978–989.

[10] P. Luo, S. Yan, Z. Liu, Z. Shen, S. Yang, and Q. He, "From online behaviors to offline retailing," in *KDD*, 2016, pp. 175–184.

[11] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, "Understanding individual human mobility patterns," *Nature*, vol. 453, no. 7196, pp. 779–782, 2008.

[12] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Scientific reports*, vol. 3, 2013.

[13] Y.-A. de Montjoye, L. Radaelli, V. K. Singh, and A. S. Pentland, "Unique in the shopping mall: On the reidentifiability of credit card metadata," *Science*, vol. 347, no. 6221, pp. 536–539, 2015.

[14] G. Salton and M. J. McGill, *Introduction to modern information retrieval*. McGraw-Hill, Inc., 1986.

[15] T.-Y. Liu, "Learning to rank for information retrieval," *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[16] T. Hao, J. Zhou, Y. Cheng, L. Huang, and H. Wu, "User identification in cyber-physical space: A case study on mobile query logs and trajectories," in *SIGSPATIAL*, 2016, pp. 71:1–71:4.

[17] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, vol. 96, no. 34, 1996, pp. 226–231.

[18] G. McLachlan and D. Peel, *Finite mixture models*. John Wiley & Sons, 2004.

[19] Z. Chen, H. T. Shen, X. Zhou, Y. Zheng, and X. Xie, "Searching trajectories by locations: An efficiency study," in *SIGMOD*, 2010, pp. 255–266.

[20] F. M. Naini, J. Unnikrishnan, P. Thiran, and M. Vetterli, "Where you are is who you are: User identification by matching statistics," *IEEE-TIFS*, vol. 11, no. 2, pp. 358–372, 2016.

[21] C. Riederer, Y. Kim, A. Chaintreau, N. Korula, and S. Lattanzi, "Linking users across domains with location data: Theory and validation," in *WWW*, 2016, pp. 707–719.

[22] M. Werner, "BACR: Set similarities with lower bounds and application to spatial trajectories," in *SIGSPATIAL*, 2015, pp. 29:1–29:10.

[23] R. T. Whitman, M. B. Park, S. M. Ambrose, and E. G. Hoel, "Spatial indexing and analytics on Hadoop," in *SIGSPATIAL*, 2014, pp. 73–82.

[24] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, "Copycatch: stopping group attacks by spotting lockstep behavior in social networks," in *WWW*, 2013, pp. 119–130.

[25] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang, "Inferring lockstep behavior from connectivity pattern in large graphs," *KAIS*, vol. 48, no. 2, pp. 399–428, 2016.

[26] T. Tian, J. Zhu, F. Xia, X. Zhuang, and T. Zhang, "Crowd fraud detection in internet advertising," in *WWW*, 2015, pp. 1100–1110.

[27] R. J. Bayardo, Y. Ma, and R. Srikant, "Scaling up all pairs similarity search," in *WWW*, 2007, pp. 131–140.

[28] A. Metwally and C. Faloutsos, "V-SMART-join: A scalable MapReduce framework for all-pair similarity joins of multisets and vectors," *PVLDB*, vol. 5, no. 8, pp. 704–715, 2012.

[29] G. Xuan, W. Zhang, and P. Chai, "EM algorithms of Gaussian mixture model and hidden Markov model," in *ICIP*, 2001, pp. 145–148.

[30] G. Schwarz *et al.*, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[31] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W.-Y. Ma, "Mining user similarity based on location history," in *SIGSPATIAL*, 2008, pp. 34:1–34:10.

[32] L. Chen and R. Ng, "On the marriage of lp-norms and edit distance," in *VLDB*, 2004, pp. 792–803.

[33] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *KDD*, 2016, pp. 785–794.

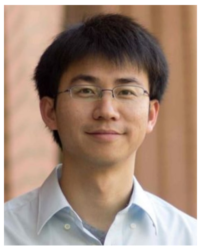
[34] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *CACM*, vol. 51, no. 1, pp. 107–113, 2008.

[35] J. Unnikrishnan and F. M. Naini, "De-anonymizing private data by matching statistics," in *Allerton Conference*, 2013, pp. 1616–1623.

Tianyi Hao Tianyi Hao is a Ph.D. candidate at the Institute for Interdisciplinary Information Sciences (IIIS) at Tsinghua University, interested in social network mining, machine learning and AI in board games. He is also working as an algorithm engineer at the Huakong Tsingjiao, focusing on federated learning. He received his B.E. degree from IIIS, Tsinghua University in 2014. He worked as an intern at the Baidu Research, focusing on spatial-temporal data mining.

Jingbo Zhou Dr. Jingbo Zhou is a staff research scientist at Business Intelligent Lab of Baidu Research, working on machine learning problems for both scientific research and business applications, with a focus on spatial temporal data mining, user behavior study and knowledge graphs. He obtained his Ph.D. degree from National University of Singapore in 2014, and B.E. degree from Shandong University in 2009. He has published several papers in top venues, such as SIGMOD, KDD, ICDE, TKDE and AAAI.

Yunsheng Cheng Yunsheng Cheng received the B.S. degree in Information Engineering and the M.S. degree in computer science from Zhejiang University.



Longbo Huang Dr. Longbo Huang is an associate professor at the Institute for Interdisciplinary Information Sciences (IIIS) at Tsinghua University, Beijing, China. Dr. Huang received the outstanding teaching award from Tsinghua university in 2014 and the ACM SIGMETRICS Rising Star Research Award in 2018. Dr. Huang currently serves as an associate editor for ACM Transactions on Modeling and Performance Evaluation of Computing Systems (ToMPECS), an editor for the IEEE Transactions on Communications (TCOM), and an associate editor for IEEE/ACM Transactions on Networking (TON). Dr. Huang's research interests are in the areas of stochastic modeling and analysis, machine learning and optimal control.

Haishan Wu Haishan Wu is currently the vice general manager of WeBank AI Group. Prior to that he was the AI scientist at BlackRock. He also worked in Baidu AI as tech leader. His research interest is using AI and big data to measure the dynamics of economic systems. He got Ph.D. degree from Fudan University in China.