International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Correlative Users travel Pattern Discovery from Smart card Data

Pathan. Ayub Khan[1], Dr. P. Siva Kumar[2]

[1, 2]Computer Science and Engineering, Rise Group of Institutions, Ongole, India

Abstract: ***Smart card transactions capture rich information of human mobility and urban dynamics, therefore are of particular interest to urban planners and location-based service providers. This provides the opportunity to discover valuable knowledge from these transaction records. In recent years, research on measuring user similarity for behaviour analysis has attracted a lot of attention in applications such as recommendation systems, crowd behavior analysis applications, and numerous data mining tasks. In this paper, our goal is to estimate the similarity between users' travel patterns according to their travel smart card data. The core of our proposal is a novel user similarity measurement, namely, Travel Spatial-Temporal Similarity (TST), which measures the spatial range and temporal similarity between users. Moreover, we also propose a hybrid index structure, which integrates inverted files and cluster-based partitioning, to allow for efficient retrieval of the top-K most similar users. Through experimental evaluation, our proposed approach is shown to deliver scalable performance.***

Keywords: smart card, spatial, temporal, partitioning, correlation

1. Introduction

The pervasive deployment of sensing technologies in cities has led to a massive increase in the volume of records of where people have been and when they were there. Those records are basically the digital footprint of individual mobility patterns. The growing use of travel smart cards is one prominent example of such technologies. Such cards facilitate the digital payment and public transport ticketing for millions of users in many metropolises. Examples include Brisbane's Go Card1, Londons Oyster Card2, and Beijing's BMAC Card3. An overview of Go Card data processing system is shown in Figure 1.

As expected, overwhelming amounts of transactional data are accumulated in the fare systems every day via those cards. Such transaction data are of particular interest to urban planners and location-based service providers, since it reveals urban dynamics and human mobility patterns. Observing and modelling human mobility pattern in urban environments are central to traffic forecasting, understanding the spread of biological viruses, designing location-based services, and improving urban infrastructure. In recent years, several efforts have been invested in mining smart card transactions leading to promising prospects in various applications such as mobility modelling and personalized recommendations [8] [19] [27].

However, in the projects mentioned above, more attention has been paid to detecting significant locations, predicting the users' movement across these locations, and recognizing the activities taking place at each location [29] [31]. Meanwhile, no significant effort has been made towards detecting the correlations between different users, though of its importance to both individuals and businesses.

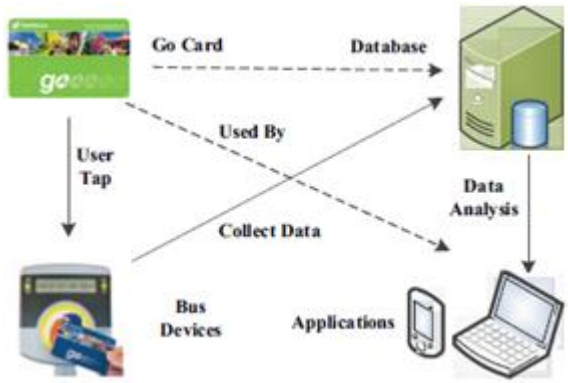

Figure 1: Data Processing system of Go Card

Orthogonally, similarity detection and measurement have been leveraged in a wide variety of applications and on various data types, such as image processing applications, trajectory data processing applications or biomedical applications, etc[17] [4] [5] [22] [20]. In addition, user similarity measurement is also a central component in many advanced applications [15] [30]. For example, recommendation systems suggest products of interest based on the similarity between users. Similarly, applications in crowd behavior analysis utilize user similarity to find common attributes among individuals. Needless to say, classical data mining tasks such as clustering and diversification problems clearly benefit from effective user similarity measures. These applications and services allow users to find others who have similar interests to learn and share information or experience. While most of those applications rely on the user's online behavior (e.g., online shopping, social networks, etc.), analyzing a user's travel behavior provides an additional wealth of valuable information. This enables extending today's applications to assist users in finding alternative travel plans and routes. It also enables businesses to provide targeted services and products that are more suitable to their customer's lifestyle, as captured by their travel patterns.

Table 1: Snapshot of Records

User Id	Boarding Station	Boarding Time	Alighting Station	Alighting Time	Bus ID
001	Garden City	15/01/2013 10:29:14	Sunnybank	15/01/2013 10:54:04	123
001	Toowong	17/01/2013 11:47:57	St Lucia	17/01/2013 12:20:13	412
002	St Lucia	15/01/2013 13:32:56	Garden City	15/01/2013 13:56:24	169
002	Garden City	16/01/2013 08:14:47	St Lucia	16/01/2013 08:40:34	169
002	Sunnybank	18/01/2013 08:22:31	Garden City	18/01/2013 08:40:15	123
003	Garden City	15/01/2013 14:34:51	St Lucia	15/01/2013 15:01:13	169
003	St Lucia	18/01/2013 18:16:47	Indooropily	18/01/2013 18:46:14	428

The work proposed in this paper is motivated by the importance of detecting user similarity in increasing the utility of large amounts of data pertaining to travel spatial-temporal patterns. In particular, in this paper, our goal is to provide efficient methods for detecting and measuring user similarity both spatially and temporally based on real world travel smart card records. In principle, our approach is a step towards mining knowledge from multiple users' spatial-temporal data. Towards achieving our goal, a similarity function, referred to as Travel Spatial-Temporal Similarity (TST), is proposed to model the similarity between users in both spatial and temporal dimensions. In addition, we address the retrieval efficiency issues relative to this similarity function. Accordingly, a new indexing framework for processing the top-K most similar user query is proposed including three pruning techniques, which are based on inverted files, cluster-based partitioning and hybrid index, to speed up the similar user exploration process.

Our key contributions in this work can be summarized as follows:

- We introduce a novel similarity function, Travel Spatial-Temporal (TST), to measure the similarity between two users. TST is based on weighted-set joins similarity and earth mover distance between histograms by quantifying the similarity between a pair of users to a value from 0 to 1. Furthermore, TST incorporates both spatial and temporal similarity together and is more robust and accurate than existing methods in measuring the similarity between two user travel profiles.
- We develop three pruning techniques - inverted files, cluster-based partitioning and hybrid index - to improve the retrieval efficiency of query. These pruning methods considers both spatial and temporal features and offer more flexibility.
- We show how to combine the pruning methods to significantly reduce the number of false positive candidates. Furthermore, we develop different variations of these pruning methods and compare their performance in terms of total computing time and pruning power.

The rest of this paper is organized as follows. Section II introduces the preliminary concepts, problem definition and overviews the similar user retrieval framework. The two core components, user similarity measurement and user retrieval index structure, are discussed in Section III and Section IV respectively. The experimental observations are presented in Section V, followed by a brief review of related work in Section VI. Section VII concludes the paper and outlines some future work.

2. Preliminaries And Problem Statement

In this section, we first clarify some of the terms and notations adopted in this work and then formally define the problem of efficiently retrieving similar users based on their travel pattern profiles, which is the main focus of this paper.

2.1 Preliminaries

Travel smart cards collect users' transaction data as they go. This data is further stored and archived in a central data warehouse. Without loss of generality, in this work we focus on the data collected and stored using Brisbane's Go Card. Typically, this raw data is stored in records according to a star schema, in which the users' transactions are stored in a fact table whereas different dimensions are represented by separate tables (e.g., user, station, etc). Table I shows a snapshot of few records from the Brisbane's Go Card database. To simplify the discussion, Table I shows those records after joining the fact table with the different dimension tables and also after eliminating some of the attributes that irrelevant to this work.

Clearly, the collected raw data, as shown in Table I, captures the different trips made by each user. In this work we define a user trip as follows:

Definition 1. (User Trip) A user trip is a journey for some purpose between a source and a destination, $P_{source} \rightarrow P_{destination}$, where p = (s; t); s is the name of the station at which the user boards or alights at, and the corresponding t is the time interval when the boarding or alighting takes place.

For example, Table I shows that user 003 made a trip, in which P_{source} = ("Garden City", 14:00 - 15:00) → $P_{destination}$ = ("St Lucia", 15:00 - 16:00) where the boarding and alighting transactions are recorded.

To capture each user's travel pattern, we propose the notion of user profile. Specifically, given a set S of all stations, a user profile captures the particular stations visited by each user, together with the visiting frequency and a time distribution of those visits. In this work, a user profile provides the basis for similarity measurement between different users and is defined as follows:

Definition 2. (User Profile) A user profile U is a set of records, such that each tuple k ∈ U is defined as(s_k, w_k, h_k), where $s_k \in$ S is a travel location descriptor in spatial space (i.e., station), w_k is the frequency of visiting s_k, and $h_k$is a histogram that describes a time distribution of those visits.

Clearly, a user profile as described above, captures the daily movement pattern of a user. In particular, given a user profile, it is easy to obtain the spatial and temporal information necessary to measure their similarity to other users. Specifically, the spatial aspect of the travel pattern is captured by the visited stations and their corresponding

frequency (i.e., s and w), whereas the temporal aspect is captured via a time distribution histogram (i.e., h), which is defined as follows.

Definition 3. (Time Distribution) A time distribution h_k is a set of records, where each record j in h_k is defined as a pair (t_i, v_j). For a station s_k, t_i is a time interval during which the user has visited s_k and v_j. is a normalized weight that is computed based on the number of times that the user has visited s_k.

In this work, we set 1 hour as the length of time interval since this is typically the time granularity used in transportation area. For example, user 002 lives near to bus station "Garden city" and goes to school everyday. Table I shows that he visited station "Garden City" during 08:00 - 09:00 for 2 times, and 13:00 - 14:00 for 1 time. Thus, we obtain his time distribution for station "Garden City" f(08:00 - 09:00, 0.67), (13:00 - 14:00, 0.33)g. Here, the weight of time interval "08:00 - 09:00" is larger than that of "13:00 - 14:00", which means the user shows up in the station more frequently during "08:00 - 09:00" than during "13:00 - 14:00".

Table II summarizes the major notations used in the rest of the paper.

Table 2: Summarize of Notations

Notation	Definition
P	Trip source or destination
U	User travel pattern profile
S	Set of stations user has visited
s_k	One station user has visited
w_k	Visiting frequency for a given s_k
h_k	User time distribution for a given s_k
t_j	Time interval user visited a station s_k
v_j	Normalized weight of visiting frequency for a given s_k and a given t_j

2.2 Problem Definition

Let D be a user profile database, in which each object is a user profile U, as described in the previous section. The main problem addressed in this paper is that given D and a user profile U_q, it is required to retrieve the Top-K users in D that are most similar to U_q. Clearly, this is equivalent to finding U_q's K Nearest Neighbor users (i.e., K-NN). Intuitively, the nearest neighbors to U_q are those whose travel range descriptors are the closest to the travel range specified in U_q, while at the same time, their travel temporal patterns are the most relevant to that of U_q. Motivated by that intuition, there are two major challenges that we address in this work:

Effectively measure the similarity between user profiles:We introduce a novel similarity function that addresses the peculiarities of travel smart card users. Given a query U_q={(s_k^q, w_k^q, h_k^q)}, and a target U_t={(s_k^t, w_k^t, h_k^t)}g, our similarity function measures the (dis)similarity between two users both spatially and temporally. By applying that function, f(U_q; U_t) returns a similarity value in [0; 1] for two users (Section III)

.

Efficiently retrieve the Top-K most similar users: We develop efficient search methods, which leverage several pruning techniques that consider both the spatial and temporal features of a user profile. By utilizing those pruning techniques, the process of exploring user profiles efficiently returns the k users with the highest similarity to query/user U_q (Section IV).

2.3 Framework Overview

Before explaining the details of our proposed methods, Figure 2 shows an overview of our proposed framework for retrieving similar users. As shown in the figure, our frameworks basically consists of three components: 1) travel pattern modelling, 2) profile indexing, and 3) similar user retrieval. The travel pattern modelling process builds a profile for every user, which is further utilized in similarity measurement. The core part of our retrieval framework is the user profile indexing. For that component, we propose to use a hybrid index structure, which integrates inverted files and cluster based partitioning, to handle both the spatial and temporal aspects of our data. Finally, in this work, we define a new similarity function TST that measures both the spatial and temporal similarity between users. In the next sections, we describe the details of each of those components.

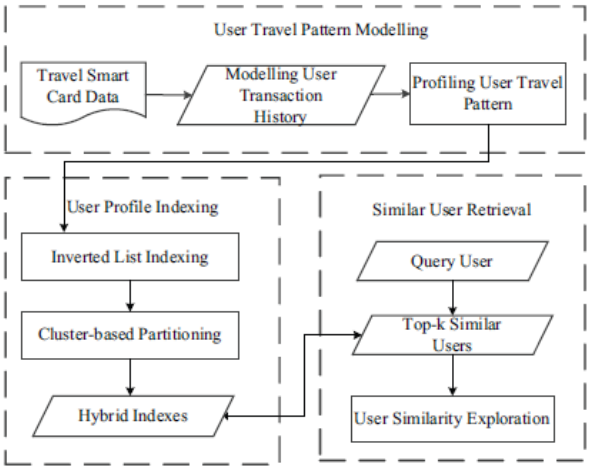

Fig. 2. Framework overview

3. Query Processing

In this section, we proceed to discuss how to retrieval similar users efficiently from a large user profile dataset. For a large scale dataset D, top-K user retrieval task returns K most similar users to a given query user q based on the previously defined similarity function. It is essential that for a given query q, we need to minimize the computation of the true distance between q and each element in D which is very time-consuming. To this end we propose a hybrid indexing structure which aims to integrate spatial relevance and temporal relevance match for user travel pattern. For spatial travel pattern, inverted files are a good choices as shown in conventional search engines. For temporal travel pattern, a cluster-based partitioning strategy is used to identify clusters from the data space. We have a strong requirement that this method causes no false dismissals while reducing false candidates. The target top-K queries return K data elements

from a database that have the greatest similarity to the query user.

3.1. Inverted Index based Search

Significant saving in computation costs can be achieved by using an index structure. For travel spatial similarity, the travel range problem can be treated as all-match selection query, which can be answered quite efficiently using an inverted file index. Recall that an all-match selection query retrieves all data strings whose similarity with the query exceeds a query threshold τ we here extend it to top-K query instead. We construct one list per station name s_i, and the list is composed of users whose travel range sets contain s_i,. Let query Sq = $\{s_1^q, s_2^q, \dots, s_n^q\}$ and each $s_i^q (1 \le i \le n)$ is assigned a weight w_i^q. Using the inverted file, we can compute TS(S_q, S_t) for all target S_t by scanning the lists of stations s_i^q . Irrelevant users (with $S_q \cap S_t \neq \emptyset$) are never accessed, this results in significant pruning compared to the brute-force approach.

Next we will discuss how to apply inverted files on top- K user retrieval which is regarded as our baseline method. Assuming that for each list i, target users St are sorting in decreasing order of $w_i^t / len(S_t)$. Notice that, len(S_q) is constant across all lists, and w_i^q is constant across list i, for a given station s_ii. Thus, the $\emptyset(w_i)$ is sorted in decreasing order for each user in corresponding list i when computing the spatial similarity. Clearly, the multi-way merge computation can be performed very efficiently if the lists are already sorted, since it compute the similarity score for every list i which is shared and determine whether the similarity is top-K. Thus, we can now use TA/NRA [7] algorithm to compute the score incrementally by using Equation (1) which is a monotonic score function.

Generally, NRA performs sequential accesses only and reads the lists in a round-robin fashion. For each list i, it iteratively loads the next element starting from the top. Since the sequential way computes similarity incrementally, it temporarily store information in main memory hash table about users whose similarity has only been partially computed so far. Each entry $S_t$of the hash table contains the aggregated score of the contributions of the lists in which S_t has already appeared. It also contains a bit vector indicating the lists where S has not been encountered yet. Figure 3 shows the sorted inverted lists and main memory hash table in processing.

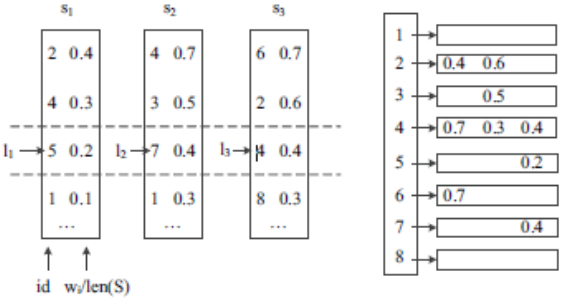

Figure 3: Inverted files and hash table

The last element on each list is denoted with li. The lower bound TS(S_q, S_t) lower is computed as the sum of the contribution$\emptyset(s_i^q)$ in each list i where S has been encountered so far. The upper bound TS(S_q, S_t))upper is computed as the sum of the lower bound and the contributions $\emptyset$ (l_i) of l_i for each list i where S has not been encountered yet. On every iteration over the lists, after all$l_i$have been updated, a top-K candidate user list is kept. we scan the target set and discard all St with upper bound smaller than the least value of current top-K candidate list. It also reports users whose score is complete and larger than the least value of current top-K candidate list, and then inserts this user into the candidate list. The algorithm here is shown in Algorithm 1, this method utilizes the early terminal condition that all the upper bounds of incomplete users is smaller than least value of current top-K candidate list, which enables processing to stop before exhaustively scanning all target users.

Algorithm 1 Using Inverted Files

Input:
Query S_q, target dataset
Output:
Top-K most similar S_t in target dataset
1: **Set** C = ∅
2: **repeat**
3: **if** S_t is complete
4: compute TS(S_q, S_t)
5: **if** TS(S_q, S_t)< min{ TS(S_q, S_p)}; $S_p \in$ C
6: insert S_t to C
7: until get top-k S_t
8: **if** TS(S_q, S_t)$_{upper}$< min{ TS(S_q, S_p)}; $S_p \in$ C
9: **if** TS(S_q, S_{t_1}) > TS(S_q, S_p); $S_p \in$ 10: insert St to C
11: return C

3.2. Cluster-based Search

For travel temporal similarity, it is important that the EMD is computed efficiently, especially for user profile dataset which is large high-dimensional dataset. Existing multi-dimensional indexes such as R-trees [9] have been shown to be inefficient for supporting range queries in high dimensional databases. Therefore, we introduce a cluster-based partitioning method to index user travel temporal histograms, which partitions the data into different clusters. In this method, The high-dimensional histograms in each cluster are treated as data points and are transformed into a single dimensional space, this allows the histograms to be retrieved using one dimensional range search.

In this method, we first partition the data points into cluster and define a reference point for each cluster. Then we index the distance of each data point to the reference point for each cluster. Without loss of generality, suppose that we obtain m clusters, $P_0, P_1, \dots . P_{m-1}$and their corresponding reference points$O_0, O_1, \dots \dots O_{m-1}$. Thus, the previously mentioned inverted file is made up of clusters and reference points instead of a bunch of data points. In the step of data partitioning, an appropriate partitioning strategy is needed to identify clusters from the data space. Here, we adopt the K-

means clustering algorithm [14] to get all the clusters and then use the center point of a cluster as reference point. Center point of each cluster is chosen as follows:

Definition 7. (Center Point of Cluster) The average distance between center point and all the other points in the cluster is smaller than that of other points.

For a given query point and a data cluster, the upper bound of similarity could be utilized to significantly reduce the amount of data that actually need to be computed by ignoring useless data points. For the scenario here, the problem of computing upper bound could be treated as finding the nearest data point in cluster to the query point. In each data cluster all data points are represented in a single dimension space.

$$y = EMD(p, O_i) \quad (5)$$

As EMD is proved to follow the triangle inequality [22] with same overall mass. Using the triangle inequality, it is straightforward to see that EMD(O_i,q) - EMD(p; q) ≤ EMD(O_i, p). When we are working with a search radius of query_ radius(q), we are interested in finding points p such that EMD(p, q) ≤query _radius(q). Letmax_i be the distance between $O_i$and the furthest point from it in partition P_i. If EMD(O_i; q) – query_ radius(q) ≤ max_i, then the nearest point in partition P_iPi to query point might locate in the overlap area which needs to be searched. The range to be searched in the single dimensional space is [EMD(O_i,q) – query_ radius(q), min(max_i;EMD(O_i; q) + query _radius(q))].

When searching for the nearest point in partition P_i to query point q, the query _radius(q) is incrementally enlarged till the first data point falls into the search "sphere". Thus, the upper bound is computed as EMD(q,$P_{nearest}$). Accordingly, given a query data point, the upper bound of each partition in the same list is computed and then sorted in descending order for later pruning. Algorithm 2 gives the illustration of cluster based partitioning algorithm utilizing distance indexing and pruning method.

3.3. Hybrid Search Algorithm

Because the two pruning methods introduced earlier are orthogonal, it is possible to combine them to constitute a hybrid search algorithm. Inverted file is used to prune search space by dismissing non-relevant data, and cluster-based approach is used to transform high-dimension data into one-dimension data for search efficiency. The hybrid method incorporates inverted file and cluster-based partition to improve pruning power. In the pre-computing procedure, inverted file is applied first, user profiles are stored in corresponding lists. Then, cluster-based partition is applied, users with similar temporal patterns in the same list are gathered together. In the query processing procedure, we sort the user cluster in descending order of the upper bound for each list. The spatial upper bound $TS(s_i)_{upper}$ of each cluster is computed by the largest value of $w_i^t / len(S_t)$. t) in cluster.

The temporal upper bound $TS(s_i)_{upper}$ of each cluster is computed by exp {-EMD(q,$P_{nearest}$)}, where $P_{nearest}$ is the nearest data point to query q. Thus, the overall upper bound $TST(s_i)_{upper} = TS(s_i)_{upper} - TT(s_i)_{upper}$. By utilizing the upper bound, top-K retrieval is processed in the previous mentioned NRA fashion. On every iteration over the lists, each cluster is considered as a computing unit. The search terminates when the upper bound of the iteration to be processed next is smaller than current least value of top-K candidate.

Algorithm 2 Using Cluster-based Partition

Input:
query point q, target dataset D
Output:
ranked point cluster
1: compute EMD(p; r), p; r ∈ D
2: partition dataset into$P_0, P_1, \ldots . P_{m-1}$
3: find center point; $O_0, O_1, \ldots \ldots O_{m-1}$
4: index points by y = EMD(p,O_i)
5: **repeat**
6: **if** EMD(O_i; q) – query_ radius(q) ≤ max_i
7: compute EMD(q; p), p ∈ P_i
8: if EMD(q; p) > query_ radius(q)
9: enlarge query_ radius(q)
10: return ranked $P_0, P_1, \ldots . P_{m-1}$

Thus, in this paper, not only do we find a good marriage between two similarity function, but we also develop an efficient hybrid search algorithm that combines two pruning strategies.

4. Conclusion And Future Work

In this paper, we have taken an important step towards effective similarity measurement of travel smart card users and efficient retrieval of top-K most similar users. We argue that an accurate and robust similarity measure is needed for searching similar users in the database, since existing ones do not handle real data well. In this paper, we propose a new similarity function, Travel Spatial-Temporal Similarity (TST), which measures the spatial range and temporal similarity between users. TST is more robust and more accurate than any existing similarity functions. After studying the spatial and temporal travel pattern similarity, we have proposed a similar user retrieval system which basically comprises three parts: travel pattern modelling, travel profile indexing and similar user retrieval

In order to improve the retrieval efficiency of TST, we propose two pruning techniques, which are inverted list and cluster-based partition, and prove that they do not introduce false dismissals by utilizing the upper bound of index structures. More importantly, we show that the two pruning methods can be combined to deliver superior retrieval efficiency.Extensive experiment have been conducted using a real travel smart card dataset, and a range of commonly used indexing methods are proposed. We have demonstrated that the TST can significantly improve both effectiveness and efficiency of similar user searching

For the future work, the ideas from this work open a new direction for future research. For example, user travel pattern similarity could be used to summarize the travel regularity of similar users group. A similarity-based travel pattern summarization framework can be proposed to incorporate with existing work, such as public transportation rescheduling and route recommendation

References

[1] Arvind Arasu, Venkatesh Ganti, and Raghav Kaushik. Efficient exact set-similarity joins. In Proceedings of the 32nd international conference on Very large data bases, pages 918–929, 2006.
[2] Xin Cao, Gao Cong, Christian S Jensen, and Beng Chin Ooi. Collective spatial keyword querying. In SIGMOD, pages 373–384, 2011.
[3] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In ICDE, pages 5–5, 2006.
[4] Lei Chen and Raymond Ng. On the marriage of lp-norms and edit distance. In VLDB, volume 30, pages 792–803, 2004.
[5] Lei Chen, M. Tamer O¨ zsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In SIGMOD, pages 491–502, 2005.
[6] Ian De Felipe, Vagelis Hristidis, and Naphtali Rishe. Keyword search on spatial databases. In IEEE 24th International Conference on Data Engineering, 2008, pages 656–665, 2008.
[7] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. Journal of Computer and System Sciences, 66(4):614 – 656, 2003.
[8] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. Nature, 453(7196):779– 782, 2008.
[9] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In SIGMOD, pages 47–57, 1984.
[10] M. Hadjieleftheriou, A. Chandel, N. Koudas, and D. Srivastava. Fast indexes and algorithms for set similarity selection queries. In ICDE, pages 267–276, 2008.
[11] Marios Hadjieleftheriou and Divesh Srivastava. Weighted set-based string similarity. IEEE Data Eng. Bull., 33(1):25–36, 2010.
[12] Ihab F Ilyas, George Beskales, and Mohamed A Soliman. A survey of top-k query processing techniques in relational database systems. ACM Computing Surveys (CSUR), 40(4):11, 2008.
[13] HV Jagadish, Beng Chin Ooi, Kian-Lee Tan, Cui Yu, and Rui Zhang. idistance: An adaptive b+-tree based indexing method for nearest neighbour search. ACM TODS, 30(2):364–397, 2005.
[14] Tapas Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. IEEE Transactions on, PAMI, 24(7):881–892, 2002.
[15] Quannan Li, Yu Zheng, Xing Xie, Yukun Chen, Wenyu Liu, and Wei-Ying Ma. Mining user similarity based on location history. In SIGSPATIAL GIS, pages 34:1–34:10, 2008.
[16] Zhisheng Li, Ken CK Lee, Baihua Zheng, Wang-Chien Lee, Dik Lun Lee, and Xufa Wang. Ir-tree: An efficient index for geographic document search. IEEE Transactions on Knowledge and Data Engineering, 23(4):585–599, 2011.
[17] Haibin Ling and K. Okada. An efficient earth mover's distance algorithm for robust histogram comparison. IEEE Transactions on PAMI, 29(5):840–853, 2007.
[18] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Machine learning, 2(4):285–318, 1988.
[19] Liang Liu, Anyang Hou, Assaf Biderman, Carlo Ratti, and Jun Chen. Understanding individual and collective mobility patterns from smart card records: A case study in shenzhen. In ITSC, pages 1–6, 2009.
[20] Ofir Pele and Michael Werman. A linear time histogram metric for improved sift matching. In ECCV, pages 495–508. 2008.
[21] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In Sixth International Conference on Computer Vision, 1998., pages 59–66, 1998.
[22] Yossi Rubner, Carlo Tomasi, and LeonidasJ. Guibas. The earth mover's distance as a metric for image retrieval. International Journal of Computer Vision, 40(2):99–121, 2000.
[23] Amit Singhal. Modern information retrieval: a brief overview. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 24, 2001.
[24] Mohamed A Soliman, Ihab F Ilyas, and K Chen-Chuan Chang. Top-k query processing in uncertain databases. In ICDE, pages 896–905, 2007.
[25] Han Su, Kai Zheng, Haozhou Wang, Jiamin Huang, and Xiaofang Zhou. Calibrating trajectory data for similarity-based analysis. In Proceedings of the 2013 international conference on Management of data, pages 833–844, 2013.
[26] Chuan Xiao, Wei Wang, Xuemin Lin, and Haichuan Shang. Top-k set similarity joins. In ICDE, pages 916–927, 2009.
[27] Yang Ye, Yu Zheng, Yukun Chen, Jianhua Feng, and Xing Xie. Mining individual life pattern based on location history. In MDM, pages 1–10, 2009.
[28] Cui Yu, Beng Chin Ooi, Kian-Lee Tan, and HV Jagadish. Indexing the distance: An efficient method to knn processing. In VLDB, volume 1, pages 421–430, 2001.
[29] Jing Yuan, Yu Zheng, and Xing Xie. Discovering regions of different functions in a city using human mobility and pois. In SIGKDD, pages 186–194, 2012.
[30] Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie, and Wei-Ying Ma. Recommending friends and locations based on individual location history. ACM Trans. Web, 5(1):5:1–5:44, 2011.
[31] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from gps trajectories. In WWW, pages 791–800, 2009.
[32] Yinghua Zhou, Xing Xie, Chuang Wang, Yuchang Gong, and Wei-Ying Ma. Hybrid index structures for location-based web search. In CIKM, pages 155–162, 2005.