Fuzzy Trajectory Linking

Huayu Wu*, Mingqiang Xue*, Jianneng Cao*, Panagiotis Karras†, Wee Siong Ng* and Kee Kiat Koo*
*Institute for Infocomm Research, A*STAR, Singapore
Email: {huwu, xuem, caojn, wsng, koo-kk}@i2r.a-star.edu.sg
†Skolkovo Institute of Science and Technology, Russia
Email: karras@skoltech.ru

***Abstract*—Today, people can access various services with smart carry-on devices, e.g., surf the web with smart phones, make payments with credit cards, or ride a bus with commuting cards. In addition to the offered convenience, the access of such services can reveal their traveled trajectory to service providers. Very often, a user who has signed up for multiple services may expose her trajectory to more than one service providers. This state of affairs raises a privacy concern, but also an opportunity. On one hand, several colluding service providers, or a government agency that collects information from such service providers, may identify and reconstruct users' trajectories to an extent that can be threatening to personal privacy. On the other hand, the processing of such rich data may allow for the development of better services for the common good.**

In this paper, we take a neutral standpoint and investigate the potential for trajectories accumulated from different sources to be *linked* so as to reconstruct a larger trajectory of a single person. We develop a methodology, called *fuzzy trajectory linking* (FTL) that achieves this goal, and two instantiations thereof, one based on hypothesis testing and one on Naïve-Bayes. We provide a theoretical analysis for factors that affect FTL and use two real datasets to demonstrate that our algorithms effectively achieve their goals.

I. Introduction

Technological developments allow for mobile and location-based services to be widely accessed by people via smart devices in urban environments on a daily basis, redefining people's lifestyles. For example, online social networking services are replacing letter communication, commuting cards are replacing paper tickets, and credit cards are replacing cash.

The systems supporting the aforementioned services are collecting rich spatiotemporal data about users' locations and the times when these users are at those locations. Taken together, the spatiotemporal data for a particular user form a trajectory. For instance, telecommunication companies maintain Call Detail Record (CDR) data of users. A cell tower's location, which is roughly the user's location, and a timestamp are recorded with each activity. It is commonplace to use commuting cards for public transportation, i.e., buses and metro, in many cities. Transportation companies and other authorities that provide such services maintain riding records, which reveal the locations and timestamps of each user's card tapings. Similarly, banks and credit card companies keep track of customer transactions, including when and where the transactions took place. Online social network platforms allow users to share their locations with their friends. The companies that operate these platforms maintain knowledge about users' whereabouts.

The above examples suggest that nowadays it is overwhelmingly difficult for people to hide their spatiotemporal data while enjoying the benefits of modern location-based services. Moreover, it is often the case that people expose their trajectories not only to a single service, but to multiple services simultaneously. A mobile phone user who exposes her data to a telecommunication provider may also be a commuter exposing her data to a transportation provider. A credit card user who exposes her data to a bank may also often use the check-in feature that exposes her locations to an online social networking platform. Therefore, if the databases of two or more service providers, whose sets of users overlap, are made available to a third party, e.g. a data analytics institute or government agency, it would be possible to link two or more trajectories of the same person using the spatiotemporal information therein.

The principle behind such trajectory linking is simple: as time goes by, the trajectories maintained by service providers grow as services are accessed by users. If two trajectories from different services belong to the same person, they should be reasonably close to each other. However, in case two trajectories belong to two different persons, they should eventually become far away from each other. We use *compatibility* to define the spatiotemporal closeness of two trajectories, which will be explained later.

We call the technology based on the above principle to link the trajectories of the same person as *fuzzy trajectory linking* (FTL). Figure 1 illustrates FTL. In this example, there are two trajectory databases. The first contains commuter trajectories generated by commuting card payments on buses and trains. Each card carries a unique ID, but may be used anonymously, i.e. the card ID is not associated with the identity of a particular person. The numbers to the left of commuter trajectories are simplified card IDs. The second database contains CDR data, which are often eponymous, as identity information usually needs to be registered when purchasing a SIM card or signing up for a mobile telephony plan. Thus, CDR trajectories are labeled by user names. If Bob, a regular mobile phone user, is also a regular commuter, FTL should be able to detect that the person carrying card ID $\#2565$ is Bob. As Figure 2 illustrates, there are two types of knowledge gained by FTL: the first is identity disclosure, as already illustrated; the second is trajectory enrichment by the merging of two linked trajectories.

With FTL, both privacy concerns and application opportunities arise. The implications on privacy arising are easy to see: an anonymous trajectory (e.g. bus riding path) of a user can be linked to an eponymous trajectory (e.g. credit card transactions) of the same user. Thus, a third party may acquire

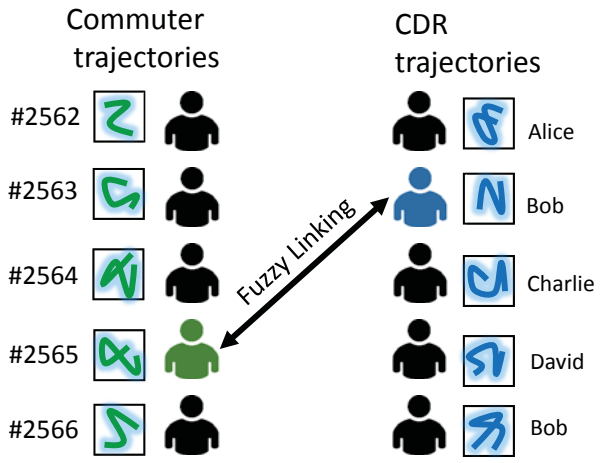


Fig. 1. A fuzzy linking example

ID	Name	Time	Location	Source
#2565	Bob	2:35pm Oct 5th	(87, 23)	CDR
		4:45pm Oct 5th	(63, 45)	Commuter
		9:10am Oct 6th	(88, 55)	CDR
		1:33pm Oct 6th	(97, 64)	CDR
		6:54pm Oct 7th	(32, 45)	Commuter

Identity disclosure | Trajectory enrichment

Fig. 2. Knowledge gain due to fuzzy linking

potentially sensitive information, e.g., learning that the user in question has been periodically visiting a hospital, a lawyer's office, or an infamous karaoke bar.

On the other hand, wisely used, FTL may inspire applications that benefit the society in the social care, crime prevention, and disease control areas. We illustrate the potentially beneficial used of FTL with two examples:

Example 1: A person is found to have contracted a serious infectious disease, such as SARS. It is known that the person has ridden several trips by bus just before found to be infected. For the purpose of disease control, the government health agency must find other commuters who have boarded the same buses. FTL can help with this task in the following way: first find the card IDs of commuters who boarded the same bus as the infected person. Then, using FTL between the trajectories of the returned card IDs and CDR data, find mobile users who have boarded on the same bus, with real identities. An FTL algorithm may return multiple mobile user matching candidates for a particular commuting card ID. However, as long as the returned number of candidates is reasonably small, the government health agency can still perform brute-force investigation over the returned candidates to prune false positives.

Example 2: Violence has erupted inside a train station. The suspect was caught on camera to have tapped his commuting card to enter the station at time 12:11pm. By searching through the riding records, it is easy to identity a set of candidate cards. However, since commuting cards are anonymous, the police will not be able to infer the identities of the owners of the candidate cards. Still, the police may seek cooperation from telecommunications providers and use FTL to match candidate trajectories against the CDR data to identify a set of possible matchings mobile users.

In this paper, we propose and study the concept of *fuzzy trajectory linking* (FTL). By this concept, we assume two databases, $\mathcal{P}$ and $\mathcal{Q}$, containing trajectories of individuals, such as the ones collected by service providers in our preceding examples. We develop two algorithms that take a subset of trajectories from $\mathcal{P}$ as queries and find a set of trajectories $\mathcal{Q}_P$ in $\mathcal{Q}$, for each query $P \in \mathcal{P}$, such that the owner of P is potentially among the owners of the trajectories in $\mathcal{Q}_P$. These two algorithms are based on hypothesis testing and Naïve-Bayes, respectively. We propose a method to rank candidates based on the likelihood of having the same owner as the query trajectory. The ranking of candidates is useful in helping the FTL user find the real matching trajectory faster. Furthermore, we analyze the factors that affect FTL in a theoretical framework.

We would like to highlight that the principle of FTL, i.e. based on the compatibilities of trajectories, does not rely on any assumption on individuals' moving patterns. The only assumption we take into account is the maximum traveling speed in a city, and we make it loose enough to accommodate all cases. For example, we assume the maximum speed V_{max} = 140 kph. Given two points in a city with geometric distance d, we only reject the cases that the traveling time between the two points is shorter than d/V_{max}. In fact, the real traveling distance is usually longer than d as no one can travel in exactly straight lines, and speed will be slower than V_{max}. Thus, our algorithm will not reject true positives. Only those point pairs that we have high confidence to deduce the violation of the maximum speed constraint will be used for justifying the compatibility of two trajectories.

Despite the simplicity of its principle, it is challenging to design FTL algorithms that work well on real data due to the reasons of:

- **Sparsity**. In real life, people may not use a service very often and thus the trajectories collected by service providers can be sparse. Thus, the techniques that we develop should not rely on properties of trajectories that only hold when the trajectory are dense. For example, when a sparse trajectory is plotted on a map, it may appear as a set of scatter points on the map instead of a continuous path, as in the case of a dense trajectory. Thus, it would be inappropriate to develop a technique based on inferring the actual moving path of a user on the map. In addition, the duration of an available data set could be short. In any case, sparsity corresponds to less information, thus creating challenges in algorithm design.
- **Non-exact matching**. People rarely use two services at the same time. Thus, the spatiotemporal records of the same person in two different databases scarcely overlap. Therefore, the algorithms we propose need to

link trajectories by more sophisticated methods than simple exact matching.

- **Inaccuracy**. In reality, the location information collected may not reflect the true locations of users. For example, location information for Location-Based Services is mostly collected by the GPS modules in users' smart phones. GPS has limited accuracy, especially in an indoor environment. For another example, the user location in CDR data is usually the location of a nearby cell tower, which can be hundreds of meters away from the real user's location. Therefore, FTL algorithms need to cope with inaccuracy in the data.

Due to these challenges, the existing trajectory similarity search algorithms, as reviewed in the next section, cannot be simply adopted to solve the FTL problem. Especially when the two trajectories exposed to two databases are sparse and have no or few location points close to each other, it is difficult to conclude the common ownership of the two trajectories based on similarity. Our proposed algorithms, instead, approach the problem from a different angle, i.e., the compatibility of data points in the two trajectories. Even if the similarity cannot be derived, our algorithms still can return a confident level on whether the two trajectories belong to the same moving object or not. At the end of paper, we show the advantage of our algorithms over similar search by experiments on real data.

The rest of the paper is organized as follows. We revisit the related work in Section II. The problem definition and relevant notations are formally introduced in Section III. Section IV is the main section of the paper presenting our algorithms to approach the FTL problem. Section V discusses the ranking method for multiple candidates. Section VI theoretically analyzes the factors that affect FTL in practice. The experimental study is presented in Section VII. Finally we conclude this paper in Section VIII.

II. Related Work

Our work is closely related to record linkage [1], which determines pairs of records referring to the same entity (e.g., an individual, a family, or an organization). Record linkage is commonly used in data integration and data cleaning (e.g., typographical error correction) and has been investigated by a large and growing body of literature. This literature can be broadly divided into approaches based on distance metrics [2], [3], [4], [5], [6] and those based on machine learning models [7], [8], [9], [10], [11].

Distance-based approaches use distance metrics and an appropriate matching threshold to find matching records. Chaudhuri et al. [2] apply edit distance to match erroneous online tuples with clean tuples from a reference relation. Guha et al. [3] propose footrule distance to merge ranking lists derived from multiple attributes, on which records are matched. Using footrule distance, they then model record matching as a minimum-cost perfect matching problem, and propose solutions to identify top-k matching records for each given record. Chaudhuri et al. [4] propose a framework for distance-based record matching. They suggest that the distance threshold for matching records be set on a case-by-case basis, instead of a global threshold. Their framework is based on two observations: *compact set*, which says that matching records are closer to each other than to other tuples, and *sparse neighborhood*, which says that the local neighborhood of matching records is sparse or empty. Other works specify distance-based rule to measure the difference between pairs of records [5], [6].

Probabilistic approaches employ machine-learning techniques such as clustering and decision trees for record linkage. TAILOR [7] is a flexible record matching toolbox, which allows users to try multiple models to match their records. This is useful, especially when users do not know which model performs most effectively on their datasets. Bhattacharya and Getoor [8] consider the relationship among records in a database (e.g., John Doe is married to Jean Doe), and perform record linkage via relational clustering. Christen [9] samples high-quality matching and non-matching pairs of records to train a classification model, based on either support vector machine (SVM) or nearest neighbor classifier. The trained model is then used to predict whether two given records match or not. Tejada et al. [10] build a committee of decision trees to find matching rules. More detailed surveys are offered in [12], [13].

Retrieval of *similar time sequences*, including trajectories [14], [15], [16], [17], [18], [19], [20], [21] is another track of research related to our work, with a wide range of applications from science to business and entertainment. Typically, it first defines a distance function, and then uses the distance function to retrieve time sequences with similar pattern to a given time sequence from a database. Representative distance functions include Euclidean distance [14], Dynamic Time Warping (DTW) [15], Longest Common Sub-Sequence (LCSS) [16], and Edit Distance on Real sequence (EDR) [17]. Agrawal et al. [14] use Discrete Fourier Transform (DFT) to map time sequences to the frequency domain, and represent each time sequence by its first few Fourier coefficients. The similarity of two time sequences is thus mapped to the Euclidean distance of their Fourier coefficients. The main drawback of this method is that the performance degrades in the presence of noise and outliers DTW [15] offers a similarity measure between two sequences that may vary in time and/or speed. LCSS [16] gives more weight to the similar portions of the sequences. Thus, it is robust to noise and variation of the sampling rate. EDR [17] is based on the Edit Distance on strings. It searches for the minimum number of edit operations (insertion, deletion, and replacement) to transform one time sequence to another. Besides the above distance functions, others include but are not limited to One Way Distance [18], Locality In-between Polylines [19], and Synchronous Euclidean Distance [20]. Wang et al. [22] make an experimental comparison of various distance functions.

To the best of our knowledge, all existing solutions of record linkage and time sequence retrieval focus on finding *similar* records or time sequences. This goal is very different from our objective, which is to determine whether two trajectories are *compatible*, while not necessarily similar. Two trajectories of the same person maintained by two different services are usually not similar, as in practice access patterns are sparse and non-overlapping.

Fuzzy trajectory linking also poses a privacy threat. We do not review the work on trajectory privacy, as it is less relevant to the contribution of this paper.

III. Definitions & Notations

Let $\mathcal{P}$ and $\mathcal{Q}$ be two trajectory databases, where each of $P \in \mathcal{P}$ and $Q \in \mathcal{Q}$ is a trajectory of a particular individual. Each trajectory is represented as a sequence of location-timestamp records sorted in time-order, i.e. $P = (p^1, p^2, \ldots, p^{|P|})$ and $Q = (q^1, q^2, \ldots, q^{|Q|})$, where p^k or q^k are the k-th time-location record of P or Q, respectively. The timestamp of a particular record p^k (q^k) is denoted as $p^j.t$ ($q^k.t$), and it follows that $p^a.t \leq p^b.t$ ($q^a.t \leq q^b.t$) if $a < b$.

We define three auxiliary functions. Given a trajectory P, $\mathsf{id}(P)$ is a function returning the identity of the owner of the trajectory, which is a unique number carried by the person such as the social security number. Given a pair of time-location records p and q, $\mathsf{dist}(p, q)$ is a function that returns the geographical distance between the location in p and the location in q. Last, $\mathsf{timediff}(p, q)$ returns the absolute difference between the timestamp in p and the timestamp in q.

Fuzzy Linking Problem Given P, a query trajectory from a trajectory database $\mathcal{P}$ and $\mathcal{Q}$, a trajectory database that contains a trajectory Q generated by the same person that generates P, i.e., $\mathsf{id}(Q) \equiv \mathsf{id}(P)$, identity a subset of trajectories $\mathcal{Q}_P \subset \mathcal{Q}$ where $|\mathcal{Q}_P| \ll |\mathcal{Q}|$, such that $Q \in \mathcal{Q}_P$.

Given a query trajectory $P \in \mathcal{P}$ and $\mathcal{Q}$, the identified candidate trajectory set $\mathcal{Q}_P$ should meet two requirements: **R1.** $\mathsf{id}(P) \equiv \mathsf{id}(Q)$, for a $Q \in \mathcal{Q}_P$, and **R2.** $|\mathcal{Q}_P| \ll |\mathcal{Q}|$.

We use these two requirements to define two performance metrics for evaluating how well an FTL algorithm compares to another. By **R1**, we define *perceptiveness*, which quantifies how good an FTL algorithm is in returning a $\mathcal{Q}_P$ that contains a trajectory belonging to user $\mathsf{id}(P)$. By **R2**, we define *selectiveness*, which quantifies the reduction in size from $|\mathcal{Q}|$ to $|\mathcal{Q}_P|$.

Definition 1: **(perceptiveness)** Given an FTL algorithm, a random $P \in \mathcal{P}$ and a $\mathcal{Q}_P$, its *perceptiveness* with respect to $\mathcal{P}$ and $\mathcal{Q}$ is $\Pr(|\{Q \,|\, Q \in \mathcal{Q}_P \wedge \mathsf{id}(P) \equiv \mathsf{id}(Q)\}| > 0)$.

Definition 2: **(selectiveness)** Given an FTL algorithm, a random $P \in \mathcal{P}$ and a corresponding $\mathcal{Q}_P$, its *selectiveness* with respect to $\mathcal{P}$ and $\mathcal{Q}$ is $\mathbb{E}\left(\frac{|\mathcal{Q}_P|}{|\mathcal{Q}|}\right)$.

A good practical criterion for selecting a good FTL algorithm is then the following: all other things being equal, an FTL algorithm that gives a larger value of *perceptiveness* is preferred over a one that gives smaller value. Yet *perceptiveness* alone is an insufficient criterion: an algorithm does nothing and returns $\mathcal{Q}$ as $\mathcal{Q}_P$ has the highest perceptiveness value. Hence the second criterion becomes handy: all other things being equal, an FTL algorithm that gives smaller value of *selectiveness* is preferred over a one that gives a larger value. In our experimental evaluation in Section VII, we use these two metrics for evaluating the proposed FTL algorithms.

IV. Algorithms

In this section we introduce two effective FTL algorithms, namely (α_1, α_2)-filtering and Naïve-Bayes-matching. At a high level, the two algorithms perform a sequence of classification tasks: they compare P with each trajectory $Q_i \in \mathcal{Q}$. If the assigned class for the pair is C_{same}, which means the two trajectories can be of the same person, the candidate Q_i is added to $\mathcal{Q}_P$; If the assigned class is C_{diff}, which means the two trajectories are of different persons, then Q_i is skipped.

Although the two algorithms work very differently in their detailed steps towards producing $\mathcal{Q}_P$, they are similar as far as both rely on a *rejection model* and an *acceptance model*, i.e., statistics we define so as to ascertain a confidence in the belief that two trajectories are of the same person. Briefly, the *rejection model* and *acceptance model* are learnt from pairs of trajectories of the same person and of different persons, respectively. The statistics we choose for each model render the two models highly distinguishable and are therefore suited for our classification task. In the following, we introduce the concepts of *trajectory alignment*, *rejection model*, and *acceptance model*; thereafter we explain in detail how our proposed algorithms use the *rejection model* and *acceptance model* to perform classification.

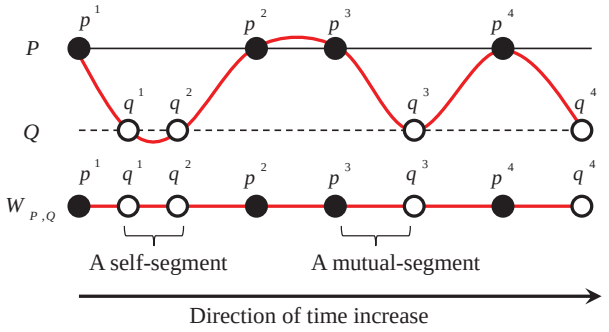


Fig. 3. Trajectory alignment, self/mutual-segment

A. Trajectory Alignment

Let $P = (p^1, p^2, \ldots, p^{|P|})$ and $Q = (q^1, q^2, \ldots, q^{|Q|})$ be two trajectories from databases $\mathcal{P}$ and $\mathcal{Q}$, respectively. The alignment of P and Q, W_{PQ}, is a trajectory $(w^1, w^2, \ldots, w^{|P|+|Q|})$ where the i-th location-timestamp record w^i is the i-th earliest record, by increasing time order, in the set $\{p^1, \ldots, p^{|P|}, q^1, \ldots, q^{|Q|}\}$. In other words, the aligned trajectory consists of location-timestamp records from both P and Q in sorted temporal order. Figure 3 shows an example of an alignment W_{PQ}, formed by aligning P and Q. An aligned trajectory can also be viewed as a sequence of segments, i.e. pairs of (w^i, w^{i+1}) where $i \in [1, |P| + |Q| - 1]$. Looking at an aligned trajectory in this perspective, we distinguish two types of segments: a *self-segment* is a segment (w^i, w^{i+1}) where both w^i and w^{i+1} are from either P or Q, whereas a *mutual segment* is a segment where one of w^i and w^{i+1} is from P and the other is from Q. In Figure 3 we label a *self-segment* by the pair (q^1, q^2) and a *mutual segment* by the pair (p^3, q^3). Thus, a *self-segment* is derived from a single trajectory, while a *mutual segment* is composed of records from different trajectories. As we explain shortly, *mutual segments* contain useful information for determining whether two trajectories belong to the same person, which we use in building our *rejection* and *acceptance* models. In the sequel, we denote $\mathsf{align}(P, Q)$ as a function that returns the aligned trajectory W_{PQ} from P and Q.

B. The Rejection-Model

In the *rejection model*, we aim to build a set of statistics that describe patterns for pairs of trajectories of the same person. We use the keyword "rejection" in the model's name because this model is used for "rejecting" a hypothesis in the α_1-rejection phase of our (α_1, α_2)-filtering algorithm. Before building our models, we need to decide the statistics they will be based on. There are two criteria for selecting appropriate statistics: (1) *availability*, which requires the statistics to be computed directly from available data, i.e. the trajectory databases $\mathcal{P}$ and $\mathcal{Q}$; and (2) *discrimination*, which requires the models to be highly distinguishable by their sets of statistics. Having these criteria in mind, we choose a set of statistics based on *mutual segment compatibility*, as defined below:

Definition 3: (**compatibility of a mutual segment**) A mutual segment (w^i, w^{i+1}) in an aligned trajectory $W_{PQ} = \mathsf{align}(P, Q)$, (w^i, w^{i+1}) is *compatible* if and only if $\frac{\mathsf{dist}(w^i, w^{i+1})}{\mathsf{timediff}(w^i, w^{i+1})} \leq V_{\mathsf{max}}$, where V_{max} is a predefined threshold for velocity. Otherwise, (w^i, w^{i+1}) is *incompatible*.

The *compatibility* of a *mutual segment* (w^i, w^{i+1}) is an evaluation of whether a person can travel from the location in w^i to the location in w^{i+1} at a reasonable speed, with reference to V_{max}. The V_{max} can either be manually set, e.g. the maximum allowed speed in a city or learnt from the data. For example, the speed of a vehicle is unlikely to be beyond 120 *kph* due to the traffic regulations; therefore if the distance between w^i and w^{i+1} is 70 *km* and the time difference is 20 minutes, it is extremely unlikely for one to have traveled from the location in w^i to the location in w^{i+1} in 20 minutes, hence this mutual segment is *incompatible*. Given an aligned trajectory $W_{P,Q}$ with *incompatible mutual segments*, a naïve reasoning would dictate that $\mathsf{id}(P) \not\equiv \mathsf{id}(Q)$. Unfortunately, such reasoning may be incorrect; due to GPS inaccuracies and software or hardware errors, real-life location-timestamp records do not accurately reflect the exact location of a person at all times. Thus, it may be that $\mathsf{id}(P) \equiv \mathsf{id}(Q)$, while $W_{P,Q}$ may still contain incompatible mutual segments. In particular, when $\mathsf{timediff}(w^i, w^{i+1})$ is small, e.g. only a few minutes, it is easy for location and/or time inaccuracies to result in incompatible mutual segments, even when $\mathsf{id}(P) \equiv \mathsf{id}(Q)$. In reverse, even if all mutual segments in $W_{P,Q}$ are compatible, this does not necessarily imply that $\mathsf{id}(P) \equiv \mathsf{id}(Q)$, since it is possible that two different persons produce compatible mutual segments, especially when $\mathsf{timediff}(w^i, w^{i+1})$ is large.

From the above discussion, it follows that we cannot directly infer whether two trajectories are of the same person or not based on the compatibility of their mutual segments alone. Besides, further contemplation leads us to the following two observations: First, given a mutual segment (w^i, w^{i+1}) from $W_{P,Q}$, the probability that (w^i, w^{i+1}) be compatible should be different in the case that $\mathsf{id}(P) \equiv \mathsf{id}(Q)$ than in the case that $\mathsf{id}(P) \not\equiv \mathsf{id}(Q)$. Second, as $\mathsf{timediff}(w^i, w^{i+1})$ affects the compatibility of mutual segments both when $\mathsf{id}(P) \not\equiv \mathsf{id}(Q)$ and when $\mathsf{id}(P) \equiv \mathsf{id}(Q)$, the probability that a mutual segment be compatible should be highly dependent on $\mathsf{timediff}(w^i, w^{i+1})$; therefore, we build our models based on the probability that mutual segments of different time lengths be compatible or incompatible, *in each of those cases*.

Note that both the rejection model and the acceptance model, described below, are built on training data. The difference in sparsity and location accuracy of different input trajectory datasets results in different parameters, e.g., V_{max} to choose, and further yields different models. This is also one of the reasons that our method is robust to different types of trajectory inputs.

The **rejection model** $M_r = \{s_r^{(1)}, \ldots, s_r^{(k)}\}$ is a set of statistics in which each $s_r^{(i)}$ is the probability that a mutual segment of i time units (after rounding to the nearest integer) be *incompatible*, when two trajectories of *the same* person are aligned.

The length of the time unit is a user-defined parameter, such as half, one, or two minutes. For example, if the time unit is one minute, $s_r^{(2)}$ is the probability that a 2-minute-long mutual segment (after rounding), in a trajectory aligning two trajectories of the same person, be incompatible. Note that M_r contains a finite number of statistics. This is so because, given enough time, one can always travel from one place to another, hence mutual segments beyond certain time difference are always *compatible*. For example, it is always feasible for one to travel from any location to any other location (via the shortest ground distance) in a big city within one hour's time assuming a speed of $V_{\mathsf{max}} = 140$ *kph*. Therefore, in this example, all mutual segments that are more than one hour long are *compatible* and their corresponding statistics 0. Such statistics are not included in M_r.

The value of a statistic s_r^i in M_r depends on the overall moving behavior of the whole users' population, which is a parameter of a complicated system. Instead of trying to obtain M_r, we can learn an estimate thereof, $\hat{M}_r$, from the trajectories in $\mathcal{P}$ and $\mathcal{Q}$, by extracting samples and feeding them to a classification algorithm. As the *rejection model* is a set of statistics for mutual segments in aligned pairs of trajectories of the same person, we need mutual segments of this kind in order to build this model. Nevertheless, we do not need to acquire real pairs of trajectories of the same person and align them. Instead, we can treat each trajectory in databases $\mathcal{P}$ and $\mathcal{Q}$ as an *already aligned* trajectory formed out of a pair of trajectories of the same person, and treat each segment in that trajectory as a mutual segment. We can do so, since any individual trajectory, just like any pair of aligned trajectories, captures the mobility of one person, and the *compatibility* of its segments is determined by the inherent inaccuracy in location and time information. Therefore, the statistics we choose for $\hat{M}_r$ satisfy the *availability* requirement. In the following, we assume the term *rejection model* refers to the estimate $\hat{M}_r$. Algorithm 1 shows the pseudo-code for building $\hat{M}_r$.

C. The Acceptance-Model

The *acceptance model* is called so as it is used for "accepting" a hypothesis in the α_2-acceptance phase of our (α_1, α_2)-filtering algorithm. The set of statistics for building the *acceptance model* is of the same kind as those for the *rejection model*, except that it is now derived from mutual segments of aligned pairs of trajectories of *different* persons. In particular, the **acceptance model** is a set of statistics $M_a = \{s_a^{(1)}, s_a^{(2)}, \ldots, s_a^{(k)}\}$, in which each $s_a^{(i)}$ is the probability that a mutual segment of i time units (after rounding to the nearest

Algorithm 1: Algorithm for building the rejection-model

Data: Trajectory databases $\mathcal{P}, \mathcal{Q}$
Result: The rejection-model $\hat{M}_r$
Let $\hat{M}_r = \{\}$ and $L = \{\}$
foreach *trajectory* t *in* $\mathcal{P} \cup \mathcal{Q}$ **do**
  foreach *segment* (w, w') *in* t **do**
    Let i = number of units in $\mathsf{timediff}(w, w')$
    Add (i, b) to L, where $b = 1$ if (w, w') is *incompatible*, or else $b = 0$
foreach *unique time-length* i *in* (i, b) *added to* L **do**
  $s_r^{(i)}$ = the percentage of $(i, 1)$ among all (i, b) in L
  add $s_r^{(i)}$ to $\hat{M}_r$
return $\hat{M}_r$

integer) be *incompatible*, when two trajectories of two *different* persons are aligned.

As we did with the rejection model, we use *acceptance model* to refer to its estimate, $\hat{M}_a$. To build $\hat{M}_a$, we need aligned pairs of trajectories of different persons. Such pairs can be obtained by selecting a trajectory $P \in \mathcal{P}$ (or $Q \in \mathcal{Q}$) and pairing it with another trajectory $P' \in \mathcal{P}$ where $P \not\equiv P'$ (or $Q' \in \mathcal{Q}$ where $Q \not\equiv Q'$). This approach is reasonable, as a user rarely has more than one trajectory in the same database. Hence, it is highly likely that $\mathsf{id}(P) \not\equiv \mathsf{id}(P')$ (or $\mathsf{id}(Q) \not\equiv \mathsf{id}(Q')$). Algorithm 2 shows the algorithm for building the *acceptance model*.

Algorithm 2: Algorithm for building the acceptance-model

Data: Trajectory databases $\mathcal{P}, \mathcal{Q}$
Result: The acceptance-model $\hat{M}_a$
Let $\hat{M}_a = \{\}$ and $L = \{\}$
foreach d *in* $\{\mathcal{P}, \mathcal{Q}\}$ **do**
  foreach *trajectory* t_a *in* d **do**
    foreach *trajectory* t_b *in* d, *s.t.* $t_a \not\equiv t_b$ **do**
      $t = \mathsf{align}(t_a, t_b)$
      foreach *mutual segment* (w, w') *in* t **do**
        Let i = number of units in $\mathsf{timediff}(w, w')$
        Add (i, b) to L, where $b = 1$ if (w, w') is *incompatible*, or else $b = 0$
foreach *unique time-length* i *in* (i, b) *added to* L **do**
  $s_a^{(i)}$ = the percentage of $(i, 1)$ among all (i, b) in L
  add $s_a^{(i)}$ to $\hat{M}_a$
return $\hat{M}_a$

D. (α_1, α_2)*-Filtering*

In a nutshell, our (α_1, α_2)-filtering algorithm starts with an initial candidate set $\mathcal{Q}_P = \mathcal{Q}$ for each trajectory $P \in \mathcal{P}$, and goes through an α_1-rejection phase and an α_2-acceptance phase, so as to filter out unqualified candidates, and leave behind the best matchings in $\mathcal{Q}_P$, where $\mathcal{Q}_P \subset \mathcal{Q}$ and $|\mathcal{Q}_P| \ll |\mathcal{Q}|$. The core of the algorithm is its filtering of unqualified candidates in its two phases. The details of the two phases are given below.

α_1**-rejection phase.**

In the α_1-rejection phase, each P is compared against each trajectory $Q \in \mathcal{Q}_P$ (originally $\mathcal{Q}_P = \mathcal{Q}$. If Q is rejected, it is removed from $\mathcal{Q}_P$; otherwise, it is kept in $\mathcal{Q}_P$ and subject to filtering in the subsequent α_2-acceptance phase. When comparing P with a particular Q, the criterion for rejecting Q as a match is based on how likely P and Q are to be of different persons. If the confidence in such nonidentity is higher than a predefined threshold, then Q is rejected. We evaluate the confidence that P and Q are of two different persons by statistical testing using our *rejection model*.

The reasoning of our statistical testing is as follows: if P and Q are indeed from the same person, then the compatibility of mutual segments in W_{PQ}, the alignment of P and Q, should be statistically compliant with the rejection model M_r. If, on the other hand, such compliance does not exist, we would rather believe that the assumption is wrong. This statistical test requires us to provide a measurement of the compliance between W_{PQ} and M_r. Such a testing statistic is the number of incompatible mutual segments in W_{PQ}. Specifically, as mutual segments in M_r have different time differences and different probabilities to be incompatible, we use a Poisson-Binomial distribution to describe the distribution of incompatible mutual segments. A Poisson-Binomial distribution is a distribution of the sum of independent Bernoulli trials of not necessarily the same success probability. The success probabilities of such Bernoulli trials are parameterized by a sequence $(p_1, p_2, \ldots, p_n)$, where p_i is the success probability of the i-th trial and n is the number of trials. In our context, the number of Bernoulli trials is the number of mutual segments in W_{PQ} and the sequence of success probabilities corresponds to $\left(s_r^{(l_1)}, s_r^{(l_2)}, \ldots, s_r^{(l_n)}\right)$, where l_i is the time length of the i-th mutual segment in W_{PQ}, n the total number of mutual segments in W_{PQ}, and $s_r^{(l_i)}$ the probability that a mutual segment of time length l_i be incompatible, according to M_r. Let K be the random variable standing for the number of incompatible mutual segments in W_{PQ}. Then we test the following hypotheses:

H_0 : K follows a Poisson-Binomial distribution parameterized by $(s_r^{(l_1)}, s_r^{(l_2)}, \ldots, s_r^{(l_n)})$
H_1 : K does not follow a Poisson-Binomial distribution parameterized by $(s_r^{(l_1)}, s_r^{(l_2)}, \ldots, s_r^{(l_n)})$

Under the *null hypothesis* H_0, the probability that a mutual segment in W_{PQ} be compatible can be derived from M_r based on the segment's time difference. In our testing, we assume the null hypothesis H_0 is true and compute the associated p-value [23], $p_{PQ}^{(\alpha_1)}$, as a function of the observed number of incompatible mutual segments in W_{PQ}, based on the following distribution function of K:

$$\Pr(K = k) = \begin{cases} \prod_{i=1}^{n} (1 - s_r^{(l_i)}) & k = 0 \\ \frac{1}{k} \sum_{i=1}^{k} (-1)^{i-1} \Pr(K = k - i) T(i) & k > 0 \end{cases} \quad (1)$$

where $T(i) = \sum_{j=1}^{n} \left(\frac{s_r^{(l_j)}}{1 - s_r^{(l_j)}} \right)^i$.

If the computed $p_{PQ}^{(\alpha_1)} < \alpha_1$, a predefined significance level, the null hypothesis is rejected. By *rejecting* the null hypothesis, we conclude that P and Q are *not* of the same person.

α_2-acceptance phase.

While the a_1-*rejection* phase is quite effective in pruning pairs of trajectories that are of different persons, not all such pairs can be pruned thereby. Thus, we design a second phase of testing, called a_2-*acceptance* phase, which reconfirms that pairs of trajectories that passed the first phase are indeed of the same person. The principle behind α_2-*acceptance* is the same as that for α_1-*rejection*, except that now the testing is based on the acceptance model M_a and corresponding hypotheses:

H_0' : K follows a Poisson-Binomial distribution parameterized by $(s_a^{(l_1)}, s_a^{(l_2)}, \ldots, s_a^{(l_n)})$

H_1' : K does not follow a Poisson-Binomial distribution parameterized by $(s_a^{(l_1)}, s_a^{(l_2)}, \ldots, s_a^{(l_n)})$

In the above, $s_a^{(l_i)}$ is the probability that the i-th mutual segment, whose time length is l_i, be incompatible according to M_a. We now use α_2 as a predefined significance level in hypothesis testing, and reject the null hypothesis H_0' when the p-value for the observed number of incompatible mutual segments in W_{PQ}, denoted as $p_{PQ}^{(\alpha_2)}$, is smaller than α_2. Since M_a is built from mutual segments in aligned trajectories of different persons, by *rejecting* the null hypothesis, we conclude that P and Q *are* of the same person. This concludes the presentation of our hypothesis testing-based methodology.

E. Naïve-Bayes-Matching

Our Naïve-Bayes-matching algorithm operates in a different manner. Let M be the random variable for the actual model, i.e., either M_a or M_r, that the observed mutual segments in W_{PQ} follow. Let $(b_1, \ldots, b_n)$ be a binary sequence, where b_i indicates the compatibility of the i-th mutual segment in W_{PQ}, with $b_i = 1$ if the segment is incompatible and $b_i = 0$ otherwise. We then evaluate, given the actual compatibilities of mutual segments $(b_1, \ldots, b_n)$, which model is more likely to be the real one. If the probability that $M = M_a$ is higher than the probability that $M = M_r$, we guess that P and Q are of two different persons; otherwise, we guess that they are of the same person. Then, we need to find the model that corresponds to $\operatorname{argmax}_M \Pr(M|(b_1, \ldots, b_n))$. By Bayes's rule, we have:

$$\begin{aligned} & \operatorname*{argmax}_M \Pr(M|(b_1, \ldots, b_n)) \\ = \; & \operatorname*{argmax}_M \frac{\Pr((b_1, \ldots, b_n)|M) \cdot \Pr(M)}{\Pr((b_1, \ldots, b_n))} \\ = \; & \operatorname*{argmax}_M \Pr((b_1, \ldots, b_n)|M) \cdot \Pr(M) \end{aligned}$$

In the above, $\Pr((b_1, \ldots, b_n)|M)$ is the probability that the mutual segments in W_{PQ} follow the observed compatibility pattern $(b_1, \ldots, b_n)$, under a given model. This probability can be computed as:

$$\Pr((b_1, \ldots, b_n)|M) \approx \begin{cases} \prod_{i=1}^{n} [s_a^{(l_i)}]^{b_i} [1 - s_a^{(l_i)}]^{1-b_i} & M = M_a \\ \prod_{i=1}^{n} [s_r^{(l_i)}]^{b_i} [1 - s_r^{(l_i)}]^{1-b_i} & M = M_r \end{cases}$$

$\Pr(M)$ is the prior probability of a model. In particular, $\Pr(M = M_r)$, denoted as ϕ_r in the sequel for simplicity, is the prior probability that a pair of trajectories be of the same person. Given two databases $\mathcal{P}$ and $\mathcal{Q}$, a fair estimate is $\phi_r \approx \frac{N_{\mathcal{PQ}}}{|\mathcal{P}| \cdot |\mathcal{Q}|}$, where $N_{\mathcal{PQ}}$ is the number of pairs of trajectories of the same person. Then, $\Pr(M = M_a)$, the prior probability that two trajectories be of different persons, denoted as ϕ_a, can be estimated as $\phi_a = 1 - \phi_r$. In practice, the value of $N_{\mathcal{PQ}}$ may not be known in advance. In this case, ϕ_r and ϕ_a can be used as weights for adjusting the strictness of the criteria in returning matching trajectories. For instance, if we wish the algorithm to loosen the criteria in returning candidate matching trajectories in $\mathcal{Q}$ for a given trajectory P, we may set a large value for ϕ_r (hence a small value for ϕ_a) so as to favor M_r as the returned model. The downside of a too large value of ϕ_r is that a large number of matching candidates may be returned for a particular query. On the other hand, if we wish to be strict in selecting matching candidates for a query, we may set a large value of ϕ_a (and hence a small value of ϕ_r), so as to favor M_a as the returned model. The downside of a too large value of $\Pr(M_a)$ is that we may not be able to find any matching candidate in $\mathcal{Q}$ for a query P. For particular applications, a user may start with a small value of ϕ_r and increase it slowly. An appropriate value of ϕ_r is one that returns a few candidate matching sets for a query, which one can prune with other approaches or investigate by brute force.

V. Ranking of Candidates

Given a query trajectory, our algorithms may return many candidate matches. Depending on parameter settings, i.e. the α_1 and α_2 values in (α_1, α_2)-filtering and the ϕ_r and ϕ_a values in Naïve-Bayes, the number of candidates may be large. Thus, it is useful to rank the candidates based on the non-increasing order of their likelihoods and examine the candidates starting from the top of the ranking. Let P be a query and Q a potentially matching candidate. In (α_1, α_2)-filtering, the p-values used for hypothesis testing (i.e., $p_{PQ}^{(\alpha_1)}$ and $p_{PQ}^{(\alpha_2)}$) provide a natural way for evaluating the likelihood of a candidate trajectory as a real matching. Specifically, in α_1-rejection, the *larger* $p_{PQ}^{(\alpha_1)}$ is, the more likely it is that P and Q are of the same person. Likewise, in α_2-acceptance, the *smaller* $p_{PQ}^{(\alpha_2)}$ is, the more likely it is that P and Q are of the same person. Then, a reasonable score to assign to any candidate Q, so as to rank candidates by the likelihood that they be true matches of P, is:

$$v_{PQ} = p_{PQ}^{(\alpha_1)}(1 - p_{PQ}^{(\alpha_2)}) \tag{2}$$

Then, in Naïve-Bayes-matching, a reasonable way to measure the likelihood that Q be a true match of P is the posterior probability $\Pr(M = M_r|(b_1, \ldots, b_n))$. The calculation of $\Pr(M = M_r|(b_1, \ldots, b_n))$ using the Bayes rule requires the prior probability $\Pr((b_1, \ldots, b_n))$. Unfortunately, $\Pr((b_1, \ldots, b_n))$ is hard to evaluate based on the training data. Besides, this value may be different for different candidates Q for a particular P; we cannot assume that it is the same for all such candidates. Nevertheless, we can use the ranking method employed for (α_1, α_2)-filtering also on the candidates returned by the Naïve-Bayes-matching. Specifically, we run

regular Naïve-Bayes-matching to identify a set of candidates Qs for a query P. Then for each candidate Q, we compute v_{PQ} as above and rank all candidates accordingly.

VI. Analysis

As we have seen, mutual segments play a very important role in FTL. Our FTL algorithms rely on the information contained in mutual segments to decide whether two trajectories are of the same person or not. Thus, the more mutual segments there are in an aligned trajectory, the easier it is for FTL algorithms to make a correct classification. Furthermore, only mutual segments within a certain time difference contain discriminating information, since a mutual segment of large time difference tends to be always *compatible* for both trajectories of the same person and different persons and thus provides little information. In this section, we study how the number and the time-difference of mutual segments are distributed under a theoretical model, in which service usage patterns follow a Poisson distribution, which is commonly used for modeling independent events that happen at a certain rate. For instance, a behavior in which a client uses one's phone 10 times per day on average, at any time of the day, can be modeled by a Poisson distribution. We model service access patterns (either by the same person or two different persons) by two independent Poisson processes N_P and N_Q, with means $\lambda_P > 0$ and $\lambda_Q > 0$, respectively. We solve three problems to find the frequency and time-length patterns of mutual segments:

Problem 1: Let X be a random variable for the number of mutual segments in a unit of time; find its probability mass function $f_X(x)$.

Although events (i.e., service accesses) are generated by a Poisson distribution N_P and N_Q, the distribution of X is not a Poisson distribution, but as follows[1]:

$$f_X(x) = \begin{cases} e^{-(\lambda_P+\lambda_Q)} \sum_{k=x+1}^{+\infty} \mu(x|k) \cdot \frac{(\lambda_P+\lambda_Q)^k}{k!} & x > 0 \\ e^{-\lambda_P} + e^{-\lambda_Q} - e^{-(\lambda_P+\lambda_Q)} & x = 0 \end{cases}$$

Where:

$$\mu(x|k) = \sum_{k'=0}^{k-x-1} (m_{x,k}(k', \lambda_P, \lambda_Q) + m_{x,k}(k', \lambda_Q, \lambda_P)) \text{ for}$$

$$m_{x,k}(k', \lambda_1, \lambda_2) = \binom{k'+\lfloor\frac{x+2}{2}\rfloor-1}{k'} \cdot \binom{k-x-1-k'+\lfloor\frac{x+1}{2}\rfloor-1}{k-x-1-k'} \cdot \frac{\lambda_1^{k'+\lfloor\frac{x+2}{2}\rfloor} \lambda_2^{k-x-1-k'+\lfloor\frac{x+1}{2}\rfloor}}{(\lambda_1+\lambda_2)^k}$$

Instead of studying $f_X(x)$ directly, we turn to the expected value of X.

Problem 2: Find the expected number of mutual segments in a unit of time, $\mathbb{E}(X)$.

The expected value of X is:

$$\mathbb{E}(X) = \frac{2\lambda_P\lambda_Q}{\lambda_P+\lambda_Q} - \left(1 - e^{-(\lambda_P+\lambda_Q)}\right) \frac{2\lambda_P\lambda_Q}{(\lambda_P+\lambda_Q)^2}$$

From this expression we can see that, for a fixed value of λ_P, $\mathbb{E}(X)$ is an increasing function of λ_Q, while $\lim_{\lambda_Q \to +\infty} \mathbb{E}(X) = 2\lambda_P$, and vice versa, $\lim_{\lambda_P \to +\infty} \mathbb{E}(X) = 2\lambda_Q$. Therefore,

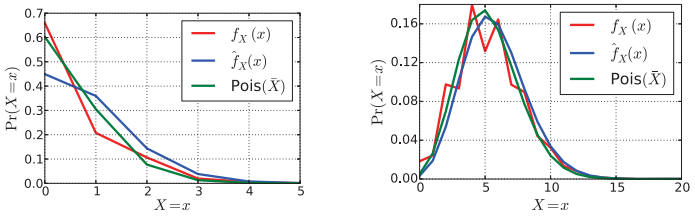
Corollary 6.1: The number of mutual segments in a unit time is bounded by $2\lambda'$, where $\lambda' = \min(\lambda_P, \lambda_Q)$.

The second term in the expression of $\mathbb{E}(X)$ is always in the open interval $(0, 0.5)$ for all values of λ_P and λ_Q. Therefore, a fair approximation for $\mathbb{E}(X)$ is:

$$\hat{\mathbb{E}}(X) = \mathbb{E}(X) + \epsilon = \frac{2\lambda_P\lambda_Q}{\lambda_P+\lambda_Q} \text{ where } \epsilon \in (0, 0.5)$$

The approximate value $\hat{\mathbb{E}}(X)$ can also be derived more intuitively: consider a particular unit of time, in which the number of events from N_P assumes its expected value λ_P. At any time point, the probability for the next event to be from N_Q is $\gamma_Q = \frac{\lambda_P}{\lambda_P+\lambda_Q}$. Then, we expect γ_Q of the λ_P events from N_P to form mutual segments with their successor events. By the same argument, we expect $\gamma_P = \frac{\lambda_P}{\lambda_P+\lambda_Q}$ of the λ_Q events from N_Q to form mutual segments with their successor events. Therefore, an estimate for the number of mutual segments is $\lambda_P\gamma_Q + \lambda_Q\gamma_P = \frac{2\lambda_P\lambda_Q}{\lambda_P+\lambda_Q}$, which is the same as the $\hat{\mathbb{E}}(X)$ derived above. The above reasoning also suggests that the number of mutual segments that start with an event from N_P (N_Q) should approximately follow a Poisson distribution with mean $\lambda_P\gamma_Q$ ($\lambda_Q\gamma_P$). When the two processes are combined, the number of mutual segments can be approximated by a Poisson distribution of mean $\hat{\mathbb{E}}(X) = \frac{2\lambda_P\lambda_Q}{\lambda_P+\lambda_Q}$:

$$\hat{f}_X(x) = \frac{\lambda^x}{x!} e^{-\lambda}, \text{ where } \lambda = \frac{2\lambda_P\lambda_Q}{\lambda_P+\lambda_Q}$$



(a) $\lambda_P = 0.5$, $\lambda_Q = 2$ (b) $\lambda_P = 4$, $\lambda_Q = 10$

Fig. 4. $f_X(x)$ vs. Poisson distributions

In Figure 4, we show the probability mass function $f_X(x)$ itself, a Poisson distribution of exactly the same mean as $f_X(x)$, and the Poisson distribution of mean $\hat{\mathbb{E}}(X)$, $\hat{f}_X(x)$, for chosen values of λ_P and λ_Q. Visibly, the three curves are similar in trend, while $f_X(x)$ is more zigzagged than the others. $\hat{f}_X(x)$ is slightly biased towards right, due to the omitted term $-\epsilon$. However, the bias decreases as λ_P and λ_Q grow.

[1]The derivation of $f_X(x)$, as well as the expected value of X in Problem 2 and the probability density function of Y in Problem 3, can be found in the extended version: http://www1.i2r.a-star.edu.sg/∼huwu/FTL_full.pdf

	S_A	S_B	S_C	S_D	S_E	S_F	T_A	T_B	T_C	T_D	T_E	T_F
*sampling rate of $\mathcal{P}$	0.006	0.008	0.01	0.01	0.01	0.01	0.06	0.07	0.08	0.08	0.08	0.08
*sampling rate of $\mathcal{Q}$	0.08	0.08	0.08	0.08	0.08	0.08	0.06	0.07	0.08	0.08	0.08	0.08
*duration (days)	31	31	31	7	14	21	7	7	7	2	4	6
mean of $\|P\|$	154.16	205.13	255.44	68.01	124.89	178.63	53.01	61.92	70.11	18.87	42.42	63.46
stdv. of $\|P\|$	97.33	129.86	159.64	121.68	144.35	148.52	128.57	149.37	166.94	33.19	80.58	162.93
mean of timediff in P (hours)	6.13	5.0	4.05	3.63	3.73	3.78	5.59	5.0	4.83	2.83	3.8	4.74
stdv. of timediff in P (hours)	7.65	6.96	5.92	4.0	4.12	4.26	6.36	5.85	5.65	2.88	4.1	5.25
mean of $\|Q\|$	67.20	67.20	67.20	15.79	30.92	45.70	53.63	62.47	71.34	19.28	44.23	64.08
stdv. of $\|Q\|$	23.17	23.17	23.17	6.24	11.18	16.13	145.18	169.08	193.07	46.92	111.12	169.44
mean of timediff in Q (hours)	13.03	13.03	13.03	10.95	12.15	12.68	4.78	4.26	3.87	2.83	3.31	3.67
stdv. of timediff in Q (hours)	14.16	14.16	14.16	11.25	12.79	13.51	5.47	4.95	4.56	2.83	3.45	4.13

TABLE I. EXPERIMENT DATA FROM SINGAPORE TAXI AND T-DRIVE DATASETS (* FOR TUNABLE PARAMETERS)

Problem 3: Let Y be the random variable for the time length of a mutual segment; find its probability density function $g_Y(y)$.

We derive $g_Y(y)$ as:

$$g_Y(y) = (\lambda_P + \lambda_Q)e^{-(\lambda_P+\lambda_Q)y}$$

It then follows that:

Corollary 6.2: The time length of a mutual segment is exponentially distributed with the mean $\frac{1}{\lambda_P+\lambda_Q}$.

Our analysis reveals the relationship between service access patterns and mutual segments. This is useful in evaluating the feasibility of FTL when real values for λ_P and λ_Q are known.

VII. EXPERIMENTAL EVALUATION

In this section, we conduct experiments to evaluate the proposed algorithms with real data. We first evaluate the effectiveness of the proposed FTL algorithms by two performance metrics proposed earlier, i.e., *perceptiveness* and *selectiveness*, as well as the effectiveness of the ranking method and the runtime efficiency of the algorithms. In the second part, we compare our algorithm with several existing trajectory similarity measurements.

A. The Datasets

For our evaluation, we have used the following real-world data.

- **Singapore Taxi Dataset.** This dataset consists of 1 month of taxi trajectories in Aug of 2008 from a major taxi company in Singapore. The data were originally stored in two databases. The first database stores log data, which consists of records reported periodically by each taxi when it is in service. Each log record contains the taxi ID, the current location, and a timestamp[2]. The time interval between two consecutive log records varies from seconds to minutes. The second database stores trip data, which consists of records reported when a taxi starts or ends a trip. A trip record contains the taxi ID, start time, start location, end time, and end location. The number of trips recorded by a taxi in a day reflects the number of served customers. For each trip record, we only consider the start time and location. The number of taxis in the log data and trip data are 15,061 and 15,028, respectively. For a particular taxi, the update frequency in log data is much denser than that in trip data.
  We would like to emphasize that the two databases are independent and contain few overlap in location points. That means, when a taxi reports its trip location to the trip database, it probably does not report its current status to the log database. As a result, this dataset is naturally a good source for validating FTL algorithms, without manual intervention.

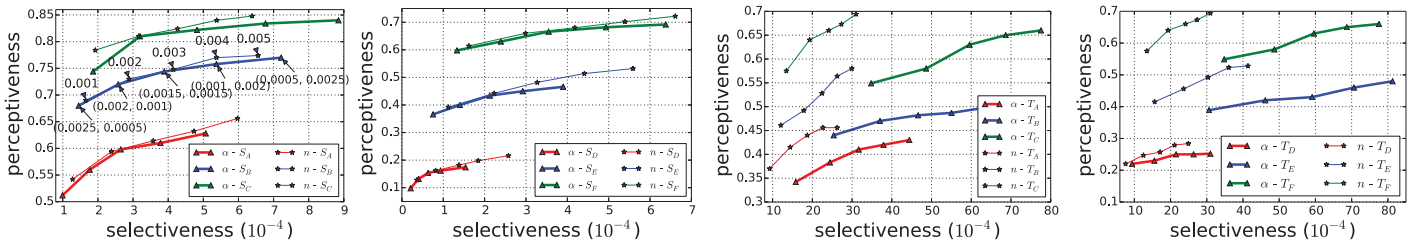
- **T-Drive Dataset**[3]**.** This datasets consists of 1 week of trajectories of 10,357 taxis in Beijing in Feb 2008. Since this dataset contains only one category of data, we manually split it into two datasets for our experiments. Each individual record is randomly dropped into one of the two datasets with the same probability. As the average sampling interval in the original T-Drive data is around 177 seconds, we expect the average sampling interval to be doubled after splitting. Again we down-sample and trim trajectories, so as to study different update frequencies and time durations.

As mentioned, the advantage of our algorithms is the handling of sparse trajectories. Thus in the first part of the experiment, to show the effectiveness of our proposed algorithms, we use high rate to down-sample the dataset to make it more sparse. By tuning the sampling rate and duration of trajectories, we form 12 different pairs of trajectory databases $\mathcal{P}$ and $\mathcal{Q}$ (Table I) from our data, where $\mathcal{P}$ is the database that generates queries and $\mathcal{Q}$ is the database that contains matching candidates. The data from S_A to S_F are derived from the Singapore taxi data. In particular, S_A, S_B, and S_C differ in sampling rate and are the same in terms of duration. Whereas, S_D, S_E, S_F are of the same sampling rate but differ in duration. Similarly, T_A to T_F are derived from the T-Drive dataset. Table I also shows important statistics of the trajectories in $\mathcal{P}$ and $\mathcal{Q}$, so as to give a better sense of the data characteristics.

In the second evaluation, to compare to other approaches, we initially make use of the Singapore taxi data in its original form, as it reflects the real situations. Then to further show the differences between algorithms in handling data sparsity, down-sampling with step-by-step sampling rate decreasements is used.

[2]Other attributes are not relevant to our evaluation, and thus omitted.

[3]http://research.microsoft.com/apps/pubs/?id=152883



(a) Diff. sampling rate S-data (b) Diff. duration S-data (c) Diff. sampling rate T-data (d) Diff. duration T-data

Fig. 5. Performance of FTL on Singapore taxi data and T-Drive data

B. Selectiveness-Perceptiveness Tradeoff

In our first experiment, we evaluate (α_1, α_2)-filtering and Naïve-Bayes-matching using the proposed evaluation metrics. An ideal algorithm should give high perceptiveness and low selectiveness values. However, it may be hard to achieve both simultaneously due to the limited quality of the data and potential ineffectiveness of the algorithm, regardless of parameter settings. Adjusting the parameters, e.g. the α_1 and α_2 values in the (α_1, α_2)-filtering, may improve one metric but sacrifice the other. Therefore, we study the tradeoff between an algorithm's perceptiveness and selectiveness under various parameter settings. Figure 5 shows the perceptiveness-selectiveness tradeoff for both (α_1, α_2)-filtering and Naïve-Bayes-matching on the Singapore taxi data (S-data) and the T-Drive data (T-data). The V_{max} that we use is 120 *kph*, a speed that taxis normally do not exceed in Singapore. For each selected parameter setting (i.e., an α_1, α_2 pair or a ϕ_r value), we randomly select 200 trajectories as queries from $\mathcal{P}$ and search for matching candidates from $\mathcal{Q}$ with our algorithms, and compute the respective perceptiveness and selectiveness. The pair of α_1 and α_2 values and the ϕ_r value that we use to create each variation are labeled in the curves that correspond to S_B in Figure 5(a). The same set of α_1 and α_2 values and ϕ_r values are used for all comparisons in Figure 5(a)-(d). Note that, based on the definitions, increasing α_1 (resp. decreasing α_2) in (α_1, α_2)-filtering, and decreasing ϕ_r in Naïve-Bayes-matching, imposes more strictness in candidate selection, and vice versa. Figures 5(a)(b) show that Naïve-Bayes-matching (curves n-S_A to n-S_F) and (α_1, α_2)-filtering (curves α-S_A to α-S_F) achieve similar tradeoffs between perceptiveness and selectiveness, with Naïve-Bayes-matching slightly outperforming (α_1, α_2)-filtering with slightly larger perceptiveness under the same selectiveness. Perceptiveness and selectiveness can be understood intuitively as follows; take a sample point on the n-S_B curve, e.g., $\phi_r = 0.005$; the corresponding selectiveness is roughly 6.5×10^{-4}, which is saying that, on average, each query (among the 200 queries selected from $\mathcal{Q}$) is matched to $15,028\times\ 6.5 \times 10^{-4} \approx 9.77$ candidates; the corresponding perceptiveness is roughly 0.77, which is saying that for 77% of the 200 query trajectories the returned candidates contain real matches. By contrast, if we had randomly selected 10 candidates, the probability that those contain a real match is only 0.06%. From Figure 5(a), we also observe that the perceptiveness for S_C is higher than for S_B, which is higher than for S_A, under the same selectiveness, for both algorithms. This result shows that higher update frequency allows for better FTL performance. Similarly, in Figure 5(b), we observe that the perceptiveness drops from S_F to S_E to S_D under the same selectiveness. This shows that trajectories of longer duration make it easier for the algorithms to find real matching candidates.

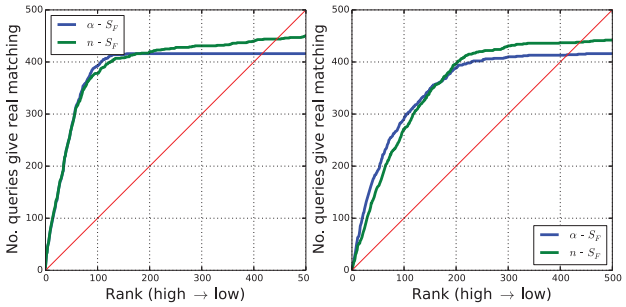
On the comparison based on T-data, shown in Figure 5(c)(d), we observe that Naïve-Bayes-matching wins by an edge compared to (α_1, α_2)-filtering. This performance gap could be due to the different characteristics between the S-data and the T-data; for example, Beijing has a much larger scale than Singapore and the duration in T-drive data tends to be much shorter than that in Singapore taxi data. In effect, the performance of both algorithms on the $S - data$ is better than their performance on the T-data, as S-data have longer duration and higher update frequency than the T-data. Among all evaluations, the worst performance occurs for the 2-day-long T_D data with (α_1, α_2)-filtering. In this setting, the returned candidates contain the real matches only for around $20 - 25\%$ of the 200 queries, while the average number of candidates for each query is around 10-30.

C. Ranking Effectiveness

In our second experiment, we evaluate the ranking effectiveness based on our two larger datasets. Recall that we can assign a score to each returned candidate, using Equation 2 obtained from the (α_1, α_2)-filtering scheme, so as to express its probability of being a true match. In this experiment, we check whether candidates with higher scores are indeed more likely to be true matches. Specifically, for the selected test data S_F and T_F, we randomly select 500 trajectories from the corresponding $\mathcal{P}$ as queries and use our algorithms to find matching candidates from the corresponding $\mathcal{Q}$. We intentionally loosen the conditions for accepting trajectories in $\mathcal{Q}$ as candidates, using $(\alpha_1, \alpha_2) = (0.001, 0.08)$ and $\phi_r = 0.4$, so that a large number of candidates is returned by each approach. We assign scores to candidates returned by both (α_1, α_2)-filtering and Naïve-Bayes-matching, based on Equation 2. We then rank candidates based on non-increasing order of ranking scores, and select the top-500 candidates from each approach. Figure 6 presents our results, as the number of queries (y-axis) for which a real match is found within the top-k candidates, where the value of k is seen along the x-axis. Observably, the growth rate of the number of queries along the y-axis slows down as the value of k grows. This result shows the effectiveness of our ranking method, as real matches tend to exist among candidates with high ranking scores.

D. Time Efficiency

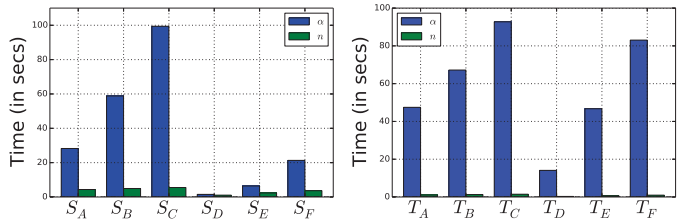
We compare the runtime efficiency of our algorithms in terms of the time needed to find matching candidates from $\mathcal{P}$



(a) Ranking on S-data (b) Ranking on T-data

Fig. 6. Candidates ranking effectiveness evaluation

for a query trajectory in from $\mathcal{Q}$. For both S-data and T-data, we randomly select 200 queries and measure the average time needed to answer one query. Figure 7 shows our results. We see that Naïve-Bayes-matching is much more efficient than (α_1, α_2)-filtering. Nevertheless, the less efficient (α_1, α_2)-filtering still answers any query in our workload within 100 seconds. We also observe that runtime naturally grows with both trajectory duration and update frequency.



(a) Run time with S-data (b) Run time with T-data

Fig. 7. Time efficiency evaluation

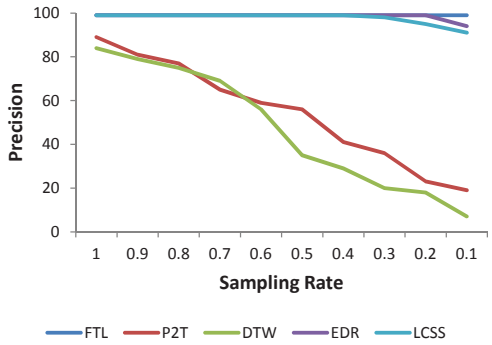
E. Comparison with Other Approaches

The most related work we identified is the trajectory similarity measurement. Intuitively, the most similar trajectories found to a query trajectory can be considered as the candidates that come from the same moving object as the query trajectory. In this section, we compare our Naïve-Bayes-matching algorithm to several well-known trajectory similarity measurement algorithms, including Point-to-trajectory (P2T), Dynamic Time Warping (DTW) [15], Longest Common Sub-Sequence (LCSS) [16], and Edit Distance on Real sequence (EDR) [17][4].

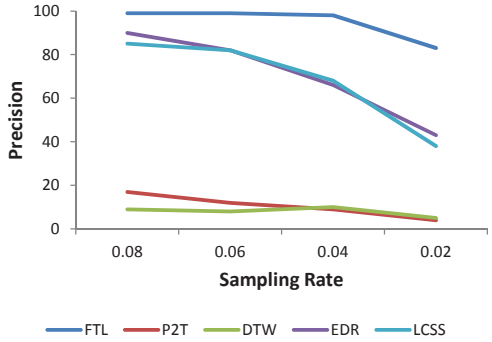
We use the Singapore taxi data for the evaluation. A query database is constructed with 100 random selected taxis in the *log* database, to search against 1000 random selected taxis (including the 100 query taxis) in the *trip* database. We do not use larger dataset because in this experiment we will require non-sampling or down-sampling with high sampling rate on the trajectories. The computation can be quite heavy, especially for those similarity search algorithms.

For all the compared algorithms, the outputs for each query are ranked by similarity values returned by the algorithm itself.

[4]The compared algorithms were implemented by Lisa Fischer and Martin Werner: http://trajectorycomputing.com/trajectory-distances-java/



(a) Down-sampling with high sampling rates



(b) Down-sampling with low sampling rates

Fig. 8. Precision comparison with other algorithms

We consider the top 10 candidates, which mean as long as the correct match is in the top 10 results, we say the algorithm can find the answer for this query. For our Naïve-Bayes-matching algorithm, we simply consider all the positive results, i.e., as long as one of them is correct, we consider the answer is found. In fact, for most of queries (over 90%), there is only one positive result. Although some queries have more than one results, the number is at most 3, which is far fewer than 10. Thus we do not need to rank the Naïve-Bayes-matching result, and the compared algorithms are actually taking advantage by using more candidates to qualify a true positive case.

Figure 8(a) shows the result with sampling rate from 1 (no sampling) to 0.1. Our algorithm is quite stable to be able to link 99 taxis[5] under all sampling rates. EDR and LCSS are also good choices, though their performance drops a little bit when the sampling rate is much lower. However, P2T and DTW have obvious performance drops as the trajectories become sparser.

Next we use even lower sampling rate to down sample the trajectories to test different algorithms. The result is shown in Figure 8(b). As expected, P2T and DTW still badly performed when the trajectories are sparse. The performance of EDR and LCSS also drops significantly when the sampling rate is less than 0.1. The robustness of our algorithm can be seen from the result. The performance of our algorithm starts dropping down after the sampling rate of 0.04, but it can still achieve a precision above 80% when the sampling rate is 0.02, i.e., only 2% of original data points are kept in each trajectory.

[5]For the one taxi our algorithm did not link, it only exposed one location point in log data, which can be viewed as a noisy trajectory.

We try to explain why the existing similarity-based algorithms cannot return good result when the trajectory is sparse. Within a city, if a trajectory only contains a few points daily (e.g., when the sampling rate is 0.02 in the Singapore taxi data, there are only one or two points reported in each hour), actually it is not a trajectory at all. We have no clue how the taxi moves, especially there are many repeated roads travelled day by day. In this case, trajectory similarity does not work well. Our approach is different. We check for compatibility between each pair of trajectories. We do not need to know whether they are similar or not given the points. That is why our approach performs much better for sparse trajectories, which is much commonly seen in real life.

For running time, our algorithm is much more efficient than others. Most other algorithms require pair-wise checking for points. Considering the high frequency of the points in the trajectories (without down-sampling or when the sampling rate is high) and long duration of the trajectories, some similarity search algorithms require days to finish a run, given the environment of a 8-core workstation with 32GB memory. We do not show the detailed running time comparison, as it is not our focus for this comparative study.

VIII. Conclusions

In this paper, we proposed a novel methodology, called *fuzzy trajectory linking*, for identifying pairs of trajectories from two different databases that belong to the same moving object, e.g., person, provided that this moving object is represented in both databases. Our proposal raises a concern for personal privacy in services that acquire user location information. At the same time, this methodology may be used in innovative applications to the benefit of society. We developed two algorithms for fuzzy trajectory linking, namely (α_1, α_2)-filtering and Naïve-Bayes-matching. We analyzed the distribution of compatible and incompatible *mutual segments*, i.e., pairs of foreign successive records in an aligned trajectory obtained by two trajectories that may or may not be of the same person; our methods draw from this analysis. We used two real datasets to evaluate the performance of the proposed algorithms and demonstrated the feasibility of FTL. Our experiments indicate that both proposed algorithms are effective in identifying real matches, with Naïve-Bayes-matching performing better than (α_1, α_2)-filtering in terms of both accuracy and runtime efficiency. Nevertheless, (α_1, α_2)-filtering is still interesting, as it provides an effective ranking mechanism for matching candidates, while its parameters are more intuitive. Furthermore, the experiments also demonstrated that the proposed algorithms is more effective than the existing similarity-based algorithms for the FTL problem.

In the future, we plan to explore parallel and distributed implementation of our algorithms for efficient large-scale fuzzy linking among several sources of trajectory data. Also, we would like to study the privacy issues brought by FTL.

Acknowledgment

This work was partially supported by Project TelePort, a joint project between Institute for Infocomm Research and Advanced Digital Sciences Center, Illinois at Singapore; and the Science and Engineering Research Council (SERC) Grant No. 152 41 00032. We would also like to thank Lisa Fischer and Martin Werner for sharing their implementations of trajectory similarity search algorithms to the research community.

References

[1] I. P. Fellegi and A. B. Sunter, "A theory for record linkage," *Journal of the American Statistical Association*, vol. 64, no. 328, pp. 1183–1210, 1969.

[2] S. Chaudhuri, K. Ganjam, V. Ganti, R. Kapoor, V. R. Narasayya, and T. Vassilakis, "Data cleaning in microsoft sql server 2005," in *SIGMOD*, 2005, pp. 918–920.

[3] S. Guha, N. Koudas, A. Marathe, and D. Srivastava, "Merging the results of approximate match operations," in *VLDB*, 2004, pp. 636–647.

[4] S. Chaudhuri, V. Ganti, and R. Motwani, "Robust identification of fuzzy duplicates," in *ICDE*, 2005, pp. 865–876.

[5] M. A. Hernández and S. J. Stolfo, "Real-world data is dirty: Data cleansing and the merge/purge problem," *Data Min. Knowl. Discov.*, vol. 2, no. 1, pp. 9–37, 1998.

[6] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita, "Declarative data cleaning: Language, model, and algorithms," in *VLDB*, 2001, pp. 371–380.

[7] M. G. Elfeky, A. K. Elmagarmid, and V. S. Verykios, "Tailor: A record linkage tool box," in *ICDE*, 2002, pp. 17–28.

[8] I. Bhattacharya and L. Getoor, "Collective entity resolution in relational data," *TKDD*, vol. 1, no. 1, 2007.

[9] P. Christen, "Automatic record linkage using seeded nearest neighbour and support vector machine classification," in *KDD*, 2008, pp. 151–159.

[10] S. Tejada, C. A. Knoblock, and S. Minton, "Learning domain-independent string transformation weights for high accuracy object identification," in *KDD*, 2002, pp. 350–359.

[11] T. Bernecker, H.-P. Kriegel, N. Mamoulis, M. Renz, and A. Züfle, "Scalable probabilistic similarity ranking in uncertain databases," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 9, pp. 1234–1246, 2010.

[12] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 1, pp. 1–16, 2007.

[13] P. Christen, "A survey of indexing techniques for scalable record linkage and deduplication," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 9, pp. 1537–1555, 2012.

[14] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases," in *FODO*, 1993, pp. 69–84.

[15] B.-K. Yi, H. V. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *ICDE*, 1998, pp. 201–208.

[16] M. Vlachos, D. Gunopulos, and G. Kollios, "Discovering similar multidimensional trajectories," in *ICDE*, 2002, pp. 673–684.

[17] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *SIGMOD Conference*, 2005, pp. 491–502.

[18] B. Lin and J. Su, "One way distance: For shape based similarity search of moving object trajectories," *GeoInformatica*, vol. 12, no. 2, pp. 117–142, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10707-007-0027-y

[19] N. Pelekis, I. Kopanakis, G. Marketos, I. Ntoutsi, G. L. Andrienko, and Y. Theodoridis, "Similarity search in trajectory databases," in *TIME*, 2007, pp. 129–140.

[20] M. Nanni and D. Pedreschi, "Time-focused clustering of trajectories of moving objects," *J. Intell. Inf. Syst.*, vol. 27, no. 3, pp. 267–289, 2006.

[21] S. Kashyap and P. Karras, "Scalable kNN search on vertically stored time series," in *KDD*, 2011, pp. 1334–1342.

[22] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. J. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Min. Knowl. Discov.*, vol. 26, no. 2, pp. 275–309, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10618-012-0250-5

[23] X. H. Chen, A. P. Dempster, and J. S. Liu, "Weighted Finite Population Sampling to Maximize Entropy," *Biometrika*, vol. 81, no. 3, 1994.