Location-Based Distance Measures for Geosocial Similarity

YARON KANZA, AT&T Labs – Research
ELAD KRAVI, Technion – Israel Institute of Technology
ELIYAHU SAFRA, General Electric Digital
YEHOSHUA SAGIV, Hebrew University

This article investigates the problem of geosocial similarity among users of online social networks, based on the locations of their activities (e.g., posting messages or photographs). Finding pairs of geosocially similar users or detecting that two sets of locations (of activities) belong to the same user has important applications in privacy protection, recommendation systems, urban planning, and public health, among others. It is explained and shown empirically that common distance measures between sets of locations are inadequate for determining geosocial similarity. Two novel distance measures between sets of locations are introduced. One is the mutually nearest distance that is based on computing a matching between two sets. The second measure uses a quad-tree index. It is highly scalable but incurs the overhead of creating and maintaining the index. Algorithms with optimization techniques are developed for computing the two distance measures and also for finding the k-most-similar users of a given one. Extensive experiments, using geotagged messages from Twitter, show that the new distance measures are both more accurate and more efficient than existing ones.

CCS Concepts: • **Information systems** → **Social networking sites**; **Location based services**; *Social recommendation*; • **Security and privacy** → *Privacy protections;*

Additional Key Words and Phrases: Geosocial networks, geospatial similarity, geosocial similarity, set distance, geotagged posts, sociospatial analysis, social media, Hausdorff distance, earth mover's distance

ACM Reference Format:
Yaron Kanza, Elad Kravi, Eliyahu Safra, and Yehoshua Sagiv. 2017. Location-based distance measures for geosocial similarity. ACM Trans. Web 11, 3, Article 17 (July 2017), 32 pages.
DOI: http://dx.doi.org/10.1145/3054951

1. INTRODUCTION

Many online social networking services and microblogging applications, including Facebook, Twitter, Foursquare, and Instagram, allow users to publish *geotagged messages* (sometimes referred to as *posts* or *tweets*). Geotagged messages indicate the location of the user at the time of the action (e.g., when posting a message or taking a picture), and thus provide information about places the user has visited.

The visited locations, as indicated by the geotagged messages, can sometimes identify a user. In other cases, they can be used for finding *geosocial similarity* (similarity, for short) between users. Intuitively, geosocial similarity between two users measures the

This research was supported in part by the Israel Science Foundation (grants 1632/12 and 1467/13) and the Israeli Ministry of Science and Technology (grant 3-9617).
Authors' addresses: Y. Kanza, AT&T Labs-Research, 1 AT&T Way, Bedminster, NJ 07921, USA; email: kanza@research.att.com; E. Kravi, Technion, Technion City, Haifa 3200003, Israel; email: ekravi@cs.technion.ac.il; E. Safra, General Electric Digital, Hamnofim 9, Herzliya, Israel; email: eli.safra@ge.com; Y. Sagiv, The Rachel and Selim Benin School of Computer Science and Engineering, The Hebrew University of Jerusalem, Rothberg Family Buildings, The Edmond J. Safra Campus, 9190416 Jerusalem, Israel; email: sagiv@cs.huji.ac.il.

DOI: http://dx.doi.org/10.1145/3054951

extent to which the locations of their messages have the same geospatial distribution—that is, it indicates whether their activities are performed in about the same locations. For instance, when two users send many messages from the same places (e.g., the same football stadium, the same concert hall, and the same train station), this increases the similarity between them. When users send messages from far apart locations, the similarity between them is deemed low.

There are different ways to use messages when trying to find similarity between users, such as by considering the content, format, and other parameters of the messages. We focus on finding similarity by merely using the locations of messages for four reasons:

(1) A location-based effective measure for detecting similarity can be combined with other methods that do consider the content of messages [Grabovitch-Zuyev et al. 2007]. In particular, we examine the extent to which message locations are sufficient for determining similarity among users.
(2) Since our methods consider only locations, they can be employed over different types of activities that are not necessarily textual and only have locations in common. For example, location-based methods can determine that two users are similar even when one posts geotagged textual messages (e.g., on Twitter) while the other uploads geotagged photos (e.g., on a photo-sharing Web site, such as Instagram).
(3) Some applications may focus on locations and thus give a greater weight to geosocial similarity compared to other types of similarity, such as applications for recommendations about locations [Ye et al. 2010; Bao et al. 2012; Wang et al. 2013] or routes [Kurashima et al. 2010; Doytsher et al. 2011].
(4) Euclidean distance or Haversine distance between points can be computed efficiently, so it is possible to use them to devise scalable distance measures for sets of locations. Taking other attributes into account might reduce efficiency and may not be applicable to large datasets.

Some works have considered the semantics of locations for measuring semantic similarity in geospatial datasets [Janowicz et al. 2008; Schwering 2008; Lee and Chung 2011]. In this article, however, we do not take the semantics of locations into account for two reasons. First, locations in our datasets are based on GPS and are not accurate enough to indicate with certainty from which building or from which floor in a building the message was sent. Second, extracting the semantics of a place requires expensive geometric operations, such as finding which polygonal area contains a given point. Our focus in this work is on efficient methods that do not require expensive computations.

Finding geosocial similarity as well as identity detection have important applications, in various domains, as elaborated next.

Applications of geosocial similarity. The following are some potential applications of geosocial similarity:

—*Sociospatial analysis*: Companies and organizations often analyze data from social media for different tasks, such as market research, learning about public opinions, inference of demographic properties of the population, and detection of trends [Carrington et al. 2005; Budak et al. 2011; Levin and Kanza 2014; Zhong et al. 2015]. Frequently, such analyses require measuring similarity or distance among users. For example, several clustering and classification methods require a notion of distance (or similarity, which is the inverse of distance) to be applied. Other advanced methods, such as blockmodeling [Doreian et al. 2005], require a notion of similarity.
—*Recommendation systems*: Similarity can be utilized in recommendation systems for suggesting to users points of interest based on locations that similar people have deemed relevant [Adomavicius and Tuzhilin 2005; Matyas and Schlieder 2009].

—*Geospatial entity resolution*: In geospatial entity resolution, the distance between different features is used to determine if two entities match [Sehgal et al. 2006]. Visits of similar users in different entities can be used, with other features, to discover matching entities.
—*Spatial outlier detection:* Given a set of similar users, detecting a small set of dissimilar users can be used for discovering outliers [Lu et al. 2003] (e.g., as part of a quality assurance procedure).

Identity detection. In identity detection [Gani et al. 2012], the goal is to determine whether two sets of geotagged messages belong to the same person. This is useful when analyzing social media data. For example, if a user publishes geotagged messages in various online social networks under different names, an identity detection could determine that all of those messages are from the same user, thereby facilitating and improving data integration.

A more important application of identity detection is de-anonymization of people who use several online social networks [Narayanan and Shmatikov 2009]—in other words, discovering that distinct users on two social media sites are actually the same person. For example, suppose that someone publishes on Twitter under the name Alice Liddell, whereas on Instagram she is known as Elizabeth Carroll. Periodically, she can test whether new activities would cause Alice Liddell to be among the k-most-similar users of Elizabeth Carroll or vice versa. If the test is positive, she knows that those new activities will breach her privacy and it is better not to carry them out. The ability to perform such a test using only locations is important for two reasons. First, posts on different social media sites may have little in common other than geospatial information. Second, as we show in this article, locations alone are quite sufficient for finding similar users. Hence, for the sake of maintaining anonymity, it is important to do a similarity test based only on locations.

As already mentioned, this article focuses on detecting similarity based only on locations of messages. Typically, similar users do not visit precisely the same places with the exact same frequency. Hence, a practical approach is to find distance measures over sets of messages that provide a good approximation of geosocial similarity. Such distance measures should satisfy the following requirements:

(1) The measure should be correlated with geosocial similarity between users.
(2) The computation of the distance should be scalable—that is, it has to be done efficiently even over many users with large sets of messages.

Our main contributions are as follows:

—We formulate geosocial similarity among users of online social networks in terms of the distance between sets of locations (assuming that they are available from users' messages). The importance of this approach was explained earlier.
—We review existing distance measures between sets of locations and explain why they are not suitable to the task of determining geosocial similarity.
—We introduce two novel distance measures that have a high correlation with geosocial similarity. One method is based on the notion of mutually nearest neighbors. We show how to compute this distance measure efficiently without preprocessing. The other method is based on quad-tree indexes and is highly efficient if those indexes are already available.
—We provide algorithms and optimization techniques for efficiently computing the k-most-similar users of a given one.
—We apply a novel methodology to test the accuracy of a distance measure for determining geosocial similarity. We show the validity of our methodology by a statistical analysis of the p-value.

—We describe extensive experiments showing that our two novel distance measures outperform existing ones in terms of both accuracy and efficiency. Moreover, our experiments also show that existing measures, such as Hausdorff distance (HD), are not appropriate for the task of detecting geosocial similarity.

The article is organized as follows. We discuss related work in Section 2. In Section 3, we present our framework and define the research problem. In Section 4, we present four traditional distance measures, for sets of points, and we illustrate and explain their disadvantages for estimating geosocial similarity. Then in Sections 5 and 6, we introduce two novel distance measures. Differently from traditional distance measures, ours are both correlated with similarity between users and can be computed efficiently. In Section 7, we study the problem of finding the k-most-similar users, for a given one, and we provide optimization techniques to enhance the efficiency of the computation. In Section 8, we present an experimental evaluation and show that our distance measures are effective in detecting similarity among users. We also show how the optimization techniques of Section 7 improve the efficiency of the computation of the k-most-similar users. We conclude in Section 9.

2. RELATED WORK

The problem of measuring similarity between users has been studied from different aspects. Cheung and Tian [2004] showed how learning techniques can be applied in recommendation systems to find similarity between users according to their preferences. A similar approach was used for utilizing user habits [Ma et al. 2012]. Several works studied the use of similarity among users in collaborative filtering [Candillier et al. 2008; Cao et al. 2008]. Anand and Bharadwaj [2010] and Stahl and Gabel [2003] showed how to apply genetic algorithms to fit similarity measures to different environments. Golbeck [2009] studied the problem of finding similarity between users in ordinary social network services (i.e., without the use of geotagging). His goal was to predict trust between users. Nisgav and Patt-Shamir [2011] studied a setting in which users are deemed similar if they provide almost identical answers to queries. McKenzie et al. [2013] showed how to apply topic modeling to use unstructured data (e.g., tips and reviews) as an additional feature when computing user similarity. Yuan et al. [2013] presented a framework for exploring and hierarchically categorizing urban lifestyles based on user activity in online social networks.

Jin et al. [2011] and Chung et al. [2014] studied the problem of finding similarity between users of online social networks based on the similarity of attributes and of the network of friends. Their goal was to cope with *identity clone attacks*. Raad et al. [2010] and Kong et al. [2013] showed how to match users in online social networks based on personal properties. An experiment that tried to match profiles of the same users in Facebook and Twitter is reported in Motoyama and Varghese [2009]. They did not use locations, and the accuracy of their approach is low relative to methods that do use the spatial locations of users. Since these works did not consider the spatial aspects of the data, our methods can strengthen their methods by adding a spatial dimension to their similarity tests.

Determining similarity between users according to locations has been studied in a few works; however, the proposed techniques were based on matching trajectories, which is very different from ours. Li et al. [2008] and Zheng et al. [2009] showed how to use data-mining techniques to find similarity between users based on the location history of the users. They showed how to extract the location history from GPS traces, construct trajectories, and find stay points of users. From the stay points and the trajectories, they build sequences of visited places, and define a matching of sequences, to measure the similarity between sequences. Similarity between users is determined according to the similarity of their sequences. Xiao et al. [2010] also investigated the

problem of determining user similarity based on GPS trajectories of the users. Their method finds the semantic meaning of the places that users have visited, and it matches the trajectories that are extracted from the location history of the users, according to the types (semantics) of the visited geographic entities. Ying et al. [2010] showed how to recommend friends based on user trajectories. Zheng and Xie [2010] studied the inverse problem of discovering similarity between locations based on visits of users in these locations. How to use MapReduce for the similarity detection was studied in Zhang et al. [2014].

The threats to the privacy of users in online geosocial networks were pointed out and stressed by several researchers [Krumm 2007; Zhong et al. 2007; Golle and Partridge 2009; Carbunar and Sion 2011; Carbunar et al. 2012; Kanza 2016]. Such threats are related to the ability to identify users based on the locations they visited and specifically linking a profile to a real person or identifying that two profiles (in a single network or in more than one network) belong to the same person. Searching for people in online social networks was studied in Motoyama and Varghese [2009]. The problem of leakage of personal information from online social networks was addressed in Krishnamurthy and Wills [2009] and Carmagnola et al. [2014].

The following two topics received a lot of attention and are somewhat related to the problem we are studying. One is spatio-textual similarity join [Ballesteros et al. 2011; Liu et al. 2012; Bouros et al. 2012; Arge et al. 1998, 2000]. The other is searches that combine text with geographic locations [Chen et al. 2006; De Felipe et al. 2008; Pat et al. 2015; Christoforaki et al. 2011; Suel 2009]. However, these topics are different from the problem we are investigating. In a similarity join, the distances are between pairs of spatial objects and not between sets of points, so the techniques are very different from those presented in this article. In a search, objects that are similar to a given query are discovered; however, also in this problem, similarity is among pairs of spatial objects and is not based on a distance between sets of points. The problem of discovering associated locations based on geotagged tweets was studied in Kanza et al. [2014], yet the association there is between locations and not between users.

In the literature, there are several distance measures for sets of points. But these measures were not designed for evaluating similarity between users, and their effectiveness for such tasks was not tested. Eiter and Mannila [1997] studied the evaluation of metric distances for sets of points. They examined five metrics and gave polynomial-time algorithms for computing them. However, their metrics are either ineffective for measuring similarity among users or require a computation that is too expensive for practical applications. Weinberger and Saul [2009] presented similarity measures for sets of points in the context of large margin nearest neighbor (LMNN) classification. Dubuisson and Jain [1994] examined variations of the HD and of the nearest-neighbor distance (NND). However, as we show in our experimental evaluation, HD and NND are ineffective for measuring spatial similarity among users.

3. PROBLEM DEFINITION

In this section, we present our framework and provide definitions and notations.

Users and messages. Many social media platforms allow users to post *geotagged messages* (*messages*, for short).[1] We denote by U the set of users. For each user $u \in U$, we denote by M_u the messages of u.

Every geotagged message is associated with a geographic location. By a slight abuse of notation, we use m to denote both a message and its location. Accordingly, $m.x$ and $m.y$ denote the x and y Cartesian coordinates, respectively, of the location of m. The

[1]For simplicity, we only consider activities of "posting a message"; however, uploading a geotagged photo or reporting the user location are equivalent to posting a message in the context of our framework.

distance between two messages, or between a message and a point, is the Haversine distance between their locations.

Geosocial similarity. Intuitively, two users are *geosocially similar* if they visit the same locations, or nearby locations, in a relatively similar frequency. In other words, if u_1 and u_2 are geosocially similar, a place that u_1 frequently visits is also frequently visited by u_2 and vice versa. A place that u_1 rarely visits (or does not visit at all) is also rarely visited (or is not visited at all) by u_2 and vice versa.

In the real world, there are no two users who are precisely the same in terms of the places they visit. Thus, similarity is determined by defining appropriate distance measures and using them inversely—a short distance indicates high similarity, whereas a long distance means low similarity.

Note that the notion of a "place" may vary in different settings and, moreover, is not always effective due to inaccuracies of locations in geotagged messages. For example, two messages that are posted from the same location may be associated with different buildings because GPS coordinates are imprecise. Yet even in such a case, the locations of the messages will be near each other. Thus, we consider locations of messages as points and base similarity detection on the distance between those points. To that end, we need to define distance measures for sets of points. In the next section, we present such measures. The distance between users is the distance between their sets of messages.

It is impossible to always identify any two similar users, because their sets of messages are never identical. This raises the issue of how to evaluate the effectiveness of a similarity measure. We propose a novel approach to this problem that is based on the observation that a user is always similar to herself. Hence, if we arbitrarily split the set of messages of a user into two, a good similarity measure should detect that those two subsets belong to the same person (i.e., the distance between them should be short). In other words, to measure the effectiveness of a distance measure, we test how well it identifies cases of two sets that belong to the same user.

The new evaluation method that we suggest can be applied in different settings. The following example illustrates it in a different scenario from the one we discuss in the rest of the article.

Example 3.1. Consider n photographers $u_1, \ldots, u_n$, each one associated with a set of artistic photos she took, and suppose that there are no signatures on the photos. Suppose that two experts, Alice and Bob, claim that they can detect similarity between artists (based on features like style, topic, etc). In such a case, it is possible to verify the claims of Alice and Bob, and to compare between them, by applying our suggested metric. In other words, the photos of some artist, say u_1, will be partitioned arbitrarily into two sets M_1 and M_2. The set M_1 will be provided to Alice and Bob, and they will need to find the set that is most similar to it among M_2 and the sets associated with $u_2, \ldots, u_n$. Deeming M_2 as the most similar to M_1 will be considered a success. Such a test could be repeated, and the experts could then be evaluated based on the number of their successes.

This test may seem to be an easy one. However, our experimental evaluation shows that when it comes to detecting similarity among users merely based on locations of activities, common methods fail.

We apply the test with respect to some given value k, as explained next.

The k-most-similar users. Given a user u, a set of users U, and a set-distance function $dist_s$, the users of U are ranked according to the distance of their messages from those of u. (We want the distance measure to be highly correlated with similarity among users;

Table I. Distance Measures and Their Notations

Distance Measure	Abbreviation	Notation
Hausdorff distance	HD	$dist_{\mathrm{h}}$
Center of mass distance (the distance between the centers of mass)	CMD_O	$dist^{o}_{cm}$
Center of mass distance (the average distance from messages of the smaller set to the center of mass of the larger set)	CMD_S	$dist^{s}_{cm}$
Center of mass distance (the average distance from messages of the larger set to the center of mass of the smaller set)	CMD_L	$dist^{l}_{cm}$
Earth mover's distance (link distance)	EMD	$dist_{em}$
Nearest-neighbor distance (the average distance from messages of the smaller set to their nearest neighbors in the larger set)	NND_S	$dist^{s}_{nn}$
Nearest-neighbor distance (the average distance from messages of the larger set to their nearest neighbors in the smaller set)	NND_L	$dist^{l}_{nn}$
Mutually nearest distance	MND	$dist_{mn}$
Quad-tree distance	QTD	$dist_{\mathrm{qt}}$

Note: MND and QTD are novel measures.

however, we can use such a measure even when it is not correlated with similarity.) The *k-most-similar users* of u are $u_1, \ldots, u_k$ in U such that for any other $u' \in U$ (i.e., u' is not one of $u_1, \ldots, u_k$), $dist_s(M_u, M_{u'}) \geq dist_s(M_u, M_{u_i})$ for all $1 \leq i \leq k$.

Effectiveness. To evaluate the *effectiveness* of a distance measure, we experimentally determine its accuracy as we now describe. First, we use two sets of messages, M and M_1, of the same user u_1. Typically, we require that $M \cap M_1 = \emptyset$. For instance, M could be the geotagged tweets of u_1 in Twitter, and M_1 could be the geotagged posts of u_1 in Instagram. Second, let $M_2, \ldots, M_n$ be the set of messages of $u_2, \ldots, u_n$, respectively, such that all u_i $(1 \leq i \leq n)$ are distinct users.

We say that M and $M_1, M_2, \ldots, M_n$ are a *success* (with respect to the parameter k) of a given set-distance function $dist_s$ if M_1 is one of the k nearest sets of M among $M_1, M_2, \ldots, M_n$. In other words, let u denote the user with the set of messages M (i.e., u and u_1 are the same user, but the former is associated with M, whereas the latter is associated with M_1). Then u_1 is one of the k-most-similar users of u among $u_1, u_2, \ldots, u_n$, that is, the users whose message sets are $M_1, M_2, \ldots, M_n$.

The *accuracy* of a set-distance function $dist_s$ is the fraction of successes in t runs, where each one comprises a different sequence of $n+1$ sets $M, M_1, M_2, \ldots, M_n$. Thus, we can compare two different distance measures based on their accuracy. Higher accuracy means higher effectiveness.

Problem definition. We have two goals: (1) to present set-distance functions that are correlated with similarity (i.e., provide high accuracy) and can be computed efficiently, and (2) to give efficient methods for finding the k-most-similar users of a given user u.

4. SIMILARITY MEASURES

Our goal is to measure similarity between users as the inverse of the distance between their message sets. To that end, we describe variations of four existing distance measures and present two new ones. The existing measures serve as baselines for evaluating the new ones. The measures and their notations are given in Table I. Mutually

nearest distance (MND) and quad-tree distance (QTD) (which are the last two rows of Table I) are the new measures that we introduce in this article. They are discussed in Sections 5 and 6, respectively. Their advantage over the traditional measures is twofold. First, they are more accurate. Second, they can be computed more efficiently.

In the definitions of all of the measures, we use $dist(m_1, m_2)$ to denote the Haversine distance between the locations of two messages m_1 and m_2. Note that in small areas, $dist(m_1, m_2)$ can be approximated by the Euclidean distance.

In this section, we describe the first seven measures of Table I, which are based on four existing methods for measuring the distance between two sets of points. We also discuss the strengths and weakness of those measures. In Sections 5 and 6, we present novel distance measures, which are more accurate and faster to compute than the traditional methods.

4.1. Hausdorff Distance

HD is a simple and common measure between sets of points [Eiter and Mannila 1997; Nutanong et al. 2011]. For each point p, we calculate the distance between p and its nearest element in the other set. Then the maximum over all of those distances is returned. We denote this distance by $dist_{\mathrm{h}}$. Formally, the HD between two sets of messages M_1 and M_2 is

$$dist_{\mathrm{h}}(M_1, M_2) = \max\left\{\max_{p_2 \in M_2}\{\min_{p_1 \in M_1}\{dist(p_1, p_2)\}\}, \max_{p_1 \in M_1}\{\min_{p_2 \in M_2}\{dist(p_1, p_2)\}\}\right\}.$$

Example 4.1. Consider the sets of messages M_1 (depicted by stars in Figure 1) and M_2 (depicted by crosses in Figure 1). The HD between the two sets is indicated by the arrow, and the effect of the removal of a single point is presented.

The HD can be useful for comparing sets of activities that are uniformly distributed in a given area [Adelfio et al. 2011]. However, there are many real-world scenarios where the HD provides poor results, because it is determined by just two points. For example, the sets of messages M_1 and M_2 in Figure 1 represent different distributions, because on the left (but not on the right) side of the area, the percentage of stars is greater than that of the crosses. Nonetheless, this does not affect the HD, whereas the removal of a single star, as illustrated in Figure 1, does change it significantly.

4.2. Center of Mass Distance

In the HD, the addition or removal of a single point can considerably change the result, even when the sets are big. A simple measure that is aimed at solving this problem is the *center of mass distance* (CMD). Given a set M_1, its *center of mass* is the location (c_x^1, c_y^1) such that c_x^1 is the average of the x coordinates and c_y^1 is the average of the y coordinates (over the locations of all messages in M_1). In other words, $c_x^1 = (\sum_{m \in M_1} m.x)/|M_1|$ and $c_y^1 = (\sum_{m \in M_1} m.y)/|M_1|$, where $|M_1|$ is the number of messages in M_1. The center of mass of M_2, denoted by (c_x^2, c_y^2), is defined similarly.

There are different variants of using the centers of mass for measuring the distance between M_1 and M_2. Next, we present three of them. The most direct way is to simply measure the Haversine distance between the two centers of mass. This measure is denoted by $dist_{cm}^{o}$ and its definition is $dist_{cm}^{o}(M_1, M_2) = dist((c_x^1, c_y^1), (c_x^2, c_y^2))$. The second way is to measure the average distance from the messages of the smaller set to the center of mass of the larger set. We denote this variant by $dist_{cm}^{s}$ and its definition is $dist_{cm}^{s}(M_1, M_2) = (\sum_{m \in M_1} dist(m, (c_x^2, c_y^2)))/|M_1|$ for $|M_1| \leq |M_2|$. The third variant, denoted by $dist_{cm}^{l}$, is to measure the average distance from the messages

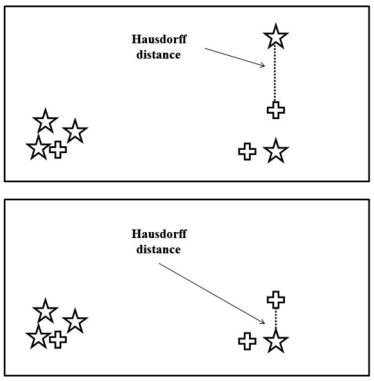

Fig. 1. The HD between two sets of points (one is shown as stars and the other as crosses). The HD is indicated by the arrow.

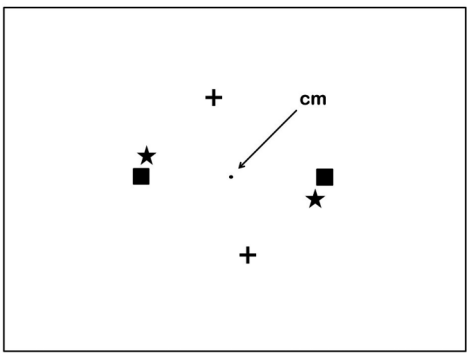

Fig. 2. Three sets with the same center of mass (cm). Each shape (square, star, cross) represents the points of a different set. The arrow indicates the center of mass of all three sets.

of the larger set to the center of mass of the smaller set—that is, $dist^{l}_{cm}(M_1, M_2) = (\sum_{m \in M_2} dist(m, (c^1_x, c^1_y)))/|M_2|$ for $|M_1| \leq |M_2|$.

Since $dist^{o}_{cm}$ only uses the centers of mass, it is less sensitive to changes than either $dist^{s}_{cm}$ or $dist^{l}_{cm}$, because the latter two take the distance to the center of mass for each message separately. Higher sensitivity in this context gives a greater weight to outliers—that is, to messages that are far from the other ones of their set. Hence, in cases where messages are distributed nonuniformly (which is common in real-world scenarios), the less sensitive method performs better than the sensitive ones. We elaborate on this in Section 8 when presenting the experimental evaluation.

Adding a single point has only a small effect on the center of mass of a large set. Thus, the (variants of the) CMD are less sensitive to small changes in the sets compared to the HD. However, the CMD does not take into account the exact distribution of points in space and therefore might not distinguish between sets of points that are very different from one another (i.e., the distance between them would be close to zero). For example, consider the three sets of messages in Figure 2. All three of them have the same center of mass (pointed to by the arrow). Indeed, two sets (the squares and the stars) are very similar to each other, but both are different from the set of crosses.

4.3. Link Distance (or Earth Mover's Distance)

The actual distribution (in space) of two sets can be effectively captured by considering a matching between their messages [Beeri et al. 2004; Safra et al. 2010]. A *matching* over M_1 and M_2 is a subset $\mu \subseteq M_1 \times M_2$ of pairs of messages (one from each set). For $(m_1, m_2) \in \mu$, we say that m_1 is *matched* with m_2 (and m_2 is matched with m_1). Note that a message of one set can be matched with several messages of the other set.

For the purpose of similarity measures, messages are matched according to their locations. There are two issues to deal with: first, how to define the distance based on a given matching, and second, which matching should be used for determining the distance. The main challenge is to apply a matching-based approach to sets of different sizes and distributions.

The link distance of Eiter and Mannila [1997] minimizes the sum of the distances between matched messages. In detail, given two sets M_1 and M_2 of messages, a matching $\mu \subseteq M_1 \times M_2$ is *complete* if every message among those of M_1 and M_2 participates in at least one pair of μ. A *minimal matching* is a complete matching μ such that

$\sum_{(m_1,m_2)\in\mu} dist(m_1, m_2) \leq \sum_{(m_1,m_2)\in\mu'} dist(m_1, m_2)$ holds for all complete matchings μ'. The *link distance* between M_1 and M_2 is the average of the distances between the pairs of a minimal matching.[2] When M_1 and M_2 have the same size, the link distance is equivalent to the known earth mover's distance (EMD).

Usually, the EMD is computed by the Hungarian algorithm, which has an $O(n^3)$ time complexity for sets of n points [Ahuja et al. 1993]. This time complexity is too high to be practical, even for sets with only a few hundred messages, assuming that there are many sets to compare. Even approximation algorithms of this problem are costly over large datasets [Hou U et al. 2008, 2010].

4.4. Nearest-Neighbor Distance

NND is a common measure. It is the average distance between a message in one set and its nearest message in the other set [Eiter and Mannila 1997].

In detail, given a message $m_1 \in M_1$, its *nearest neighbor* in M_2, denoted by $NN_{M_2}(m_1)$, is the message m_2 that has the shortest distance[3] to m_1 among all messages of M_2. In other words, $NN_{M_2}(m_1)$ is the message $m_2 \in M_2$ such that $dist(m_1, m_2) \leq dist(m_1, m')$ *for every* $m' \in M_2$. The nearest neighbor in M_1 of a message $m_2 \in M_2$ is defined similarly. The *distance-to-nearest* of m_1 is the distance from m_1 to its nearest neighbor in the other set.

As in the case of the CMD, the NND approach can be applied in different manners. We define two of them. Let M_1 and M_2 be two sets of messages such that M_1 is smaller than M_2 (i.e., $|M_1| \leq |M_2|$). We denote by $dist^s_{nn}$ the average distance-to-nearest over the messages of the smaller set—that is,

$$dist^s_{nn}(M_1, M_2) = \frac{\sum_{m_1\in M_1} dist(m_1, NN_{M_2}(m_1))}{|M_1|}.$$

We denote by $dist^l_{nn}$ the average distance-to-nearest over the messages of the larger set—that is,

$$dist^l_{nn}(M_1, M_2) = \frac{\sum_{m_2\in M_2} dist(m_2, NN_{M_1}(m_2))}{|M_2|}.$$

In cases such as the one in Figure 2, the NND (of both versions) deems that the sets of stars and squares are similar to each other, but both are different from the set of crosses. However, this approach does not deal well with clusters of messages. In Figure 4, for example, the two sets (of stars and squares) have clusters in different locations. Hence, they should be considered dissimilar; however, the NND between them is short. Furthermore, the nearest-neighbor approach cannot distinguish the case of the dissimilar sets in Figure 4 from the similar ones in Figure 3, because in the former each square has a nearby star and vice versa, so the distance to the nearest neighbor is short.

Thus far, we have discussed common methods or variations thereof. In the next two sections, we present two novel distance measures and illustrate their advantages over the existing ones.

5. MUTUALLY NEAREST DISTANCE

In this section, we present MND. We show how to compute it efficiently and compare it to the measures discussed earlier.

[2]Traditionally, sum is applied to the pairwise distances, instead of average. However, the sum of distances between matched messages is problematic, because it is affected by the sizes of the sets.

[3]We make sure that the nearest neighbor of an element is always unique. If needed, we do so by slightly modifying the distance function.

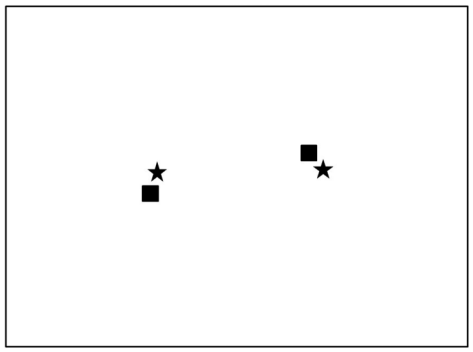

Fig. 3. Two similarly distributed sets.

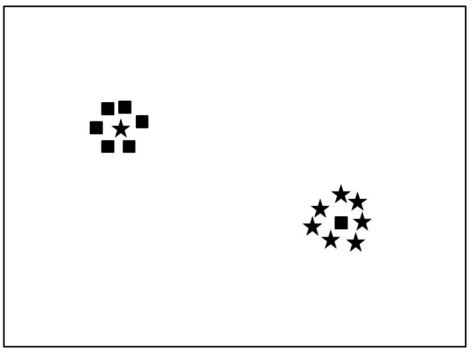

Fig. 4. Two dissimilar clustered sets.

As already shown, the nearest-neighbor approach sometimes uses a matching that is not suitable for defining the distance between sets of points. MND resembles NND in the sense that it measures the average distance between matched pairs of messages. However, MND employs a matching that is defined differently from the one used by NND. Next, we define this matching and explain how to use it for measuring the distance between sets of messages.

Definition 5.1 (*Mutually Nearest*). Given two messages $m_1 \in M_1$ and $m_2 \in M_2$, we say that m_1 and m_2 are *mutually nearest* if m_1 is the nearest neighbor of m_2 in M_1 and m_2 is the nearest neighbor of m_1 in M_2.

Given two sets of equal size, a *mutually nearest matching* is iteratively constructed as follows. Each iteration matches the pair of mutually nearest messages that have the shortest distance between them. The matched messages are deleted, and the process continues with the remaining messages. Since the two sets are of equal size, this process creates a one-to-one matching between their messages.

Mutually nearest matching is suitable for measuring similarity because each point participates in the matching and affects the measure proportionally to its weight in its containing set—that is, in a set of n points, a single point has a weight of $\frac{1}{n}$. For comparison, we have seen that in the HD, a single point may significantly affect the distance. Additionally in the nearest-neighbor method, where several points can be matched to a single point, the effect of a single point on the measure can be much greater than the effect of other points from the same set.

5.1. Dealing with Sets of Unequal Size

Next, we describe how to construct a mutually nearest matching when the sets are of unequal size. In this case, we have to match a message of the smaller set with several ones from the larger set. For example, if there are two messages in the smaller set M_1 and four messages in the larger set M_2, then each message of M_1 is matched with two messages of M_2. However, we need to prevent a case where we give greater weights to some messages than to others, which could happen when the ratio $\frac{|M_2|}{|M_1|}$ of the numbers of messages in the two sets is not an integer. For example, suppose that there are two messages in M_1 and three messages in M_2. If one message of M_1 appears in a single pair of the matching, whereas the other appears in two of them, then the average distance between pairs may depend on the frequency of messages in the matching. Alternatively, if we match each message of M_1 with only one message of M_2, then some messages of M_2 will not be included in the matching.

To solve this problem, we create two new sets from M_1 and M_2 as follows. M_1^* comprises $|M_2|$ copies of each message of M_1, and M_2^* comprises $|M_1|$ copies of every message of M_2. Each copy is a new distinct message with the same location as the original message. Note that each one of the sets M_1^* and M_2^* comprises $|M_1| \cdot |M_2|$ messages. For example, if there are two messages in M_1 and three messages in M_2, we create three copies of the messages of M_1 and two copies of the messages of M_2, so there are six messages in each of M_1^* and M_2^*. (To satisfy the requirement that the sets M_1^* and M_2^* are of equal size, we could reduce the number of copies by using the least common multiple of $|M_1|$ and $|M_2|$. However, this affects neither the complexity of the computation nor the quality of the matching, but it slightly complicates the description.)

To construct the mutually nearest matching μ over M_1^* and M_2^*, we also use the original sets M_1 and M_2 and do it as follows. Let (m_1, m_2), where $m_1 \in M_1$ and $m_2 \in M_2$, be a pair of mutually nearest messages with the shortest distance. Let m_1^* and m_2^* be copies of m_1 and m_2 in M_1^* and M_2^*, respectively. Then (1) (m_1^*, m_2^*) is added to μ; (2) m_1^* and m_2^* are discarded from M_1^* and M_2^*; and (3) when there are no more copies of m_1 in M_1^*, the message m_1 is removed from M_1, and the same is applied to m_2 and M_2. This process terminates when the sets M_1, M_2, M_1^*, and M_2^* are empty. The four sets become empty at the same time for the following reason. Initially, the sets M_1^* and M_2^* have the same number of elements. In each iteration, their sizes decrease by one. In addition, the last element of M_1 is removed when M_1^* becomes empty, and the same holds for M_2 and M_2^*.

The preceding process is well defined because there is always at least one pair of mutually nearest messages in M_1 and M_2, as stated by the following proposition. We assume that each pair of messages (not necessarily mutually nearest) has a unique distance. In other words, for every $m_1, m_1' \in M_1$ and $m_2, m_2' \in M_2$, if $(m_1, m_2) \neq (m_1', m_2')$, then $dist(m_1, m_2) \neq dist(m_1', m_2')$. (Note that $(m_1, m_2) \neq (m_1', m_2')$ if either $m_1 \neq m_1'$, $m_2 \neq m_2'$ or both.)

PROPOSITION 5.2. *Consider two nonempty sets of messages M_1 and M_2, such that distances between pairs are unique. Then there exists a pair $(m_1, m_2) \in M_1 \times M_2$ such that m_1 and m_2 are mutually nearest.*

Proposition 5.2 is actually a corollary of Proposition 5.3, which also shows the following. We can efficiently find the pair of mutually nearest messages such that the distance between them is the shortest.

PROPOSITION 5.3. *Consider two nonempty sets of messages M_1 and M_2 such that distances between pairs are unique. Let $(m_1, m_2) \in M_1 \times M_2$ have the shortest distance among all pairs of $M_1 \times M_2$. In other words, $dist(m_1, m_2) < dist(m_1', m_2')$ for all $(m_1', m_2') \neq (m_1, m_2)$ in $M_1 \times M_2$. Then m_1 and m_2 are mutually nearest. Hence, (m_1, m_2) has the shortest distance among all pairs of mutually nearest messages.*

PROOF. If m_1 and m_2 are not mutually nearest, then either there is a message m_2'' in M_2 that is closer to m_1 than m_2 or there is a message m_1'' in M_1 that is closer to m_2 than m_1. In the first case, $dist(m_1, m_2'') < dist(m_1, m_2)$ and in the second case $dist(m_1'', m_2) < dist(m_1, m_2)$, in contradiction to the assumption that (m_1, m_2) is the pair with the shortest distance between its messages. Thus, m_1 and m_2 are mutually nearest. Since the distance between m_1 and m_2 is the shortest among all pairs of $M_1 \times M_2$, it is also the shortest among all pairs of mutually nearest messages. □

Consider again the iterative process described earlier for computing the mutually nearest matching μ. According to Proposition 5.3, to find the required pair in each iteration, we only need to choose the one that has the shortest distance between its messages among all existing pairs in $M_1 \times M_2$.

The removal of messages can be done in batches. In other words, suppose that (m_1, m_2) is the pair of mutually nearest messages with the shortest distance. Let n_1 and n_2 be the number of copies of m_1 and m_2 in M_1^* and M_2^*, respectively. If $n_1 \leq n_2$, then in one iteration we add n_1 pairs of (m_1^*, m_2^*) to μ, discard n_1 copies of m_1 and m_2 from M_1^* and M_2^*, respectively, and remove m_1 from M_1; in addition, if $n_1 = n_2$, then we also remove m_2 from M_2. If $n_1 > n_2$, we add n_2 copies of (m_1^*, m_2^*) to μ, update M_1^* and M_2^*, and remove m_2 from M_2. Note that in each iteration we remove an element of M_1, remove an element of M_2, or both.

The created matching can be viewed as a *fractionated matching*, where the matching of a message (of one set) is partitioned between two or more messages (of the other set). For instance, a message m_1 may have n copies such that $\frac{x}{n}$ of the copies are associated with message m_2 and $\frac{n-x}{n}$ of the copies are associated with message m_2'.

5.2. Sorting Enhances Efficiency

There are at most $|M_1||M_2|$ pairs (since messages are only removed from the two sets during the process). If in each iteration we traverse the whole list of pairs to find the one with the minimum distance, then computing μ takes $O(|M_2||M_1||M_2|)$ time, assuming that $|M_1| \leq |M_2|$, because there are at most $|M_1| + |M_2|$ iterations. Alternatively, we can initially sort (in ascending order) all $|M_1||M_2|$ pairs by their distance. Now we have to traverse the sorted list just once. In the ith iteration, we choose the next eligible pair in sorted order—that is, the next pair such that neither of its messages has yet been removed. Assuming that we can test eligibility of a pair in constant time, the whole process takes $O(|M_1||M_2| \log(|M_1||M_2|))$ time. Thus, computing μ in this way is more efficient than an evaluation that does not sort the pairs, provided that $\log(|M_1||M_2|) \leq |M_2|$. To see why this is true, note that $\log(|M_1||M_2|) \leq \log(|M_2||M_2|)$ because $|M_1| \leq |M_2|$. Basic properties of logarithms imply that $\log(|M_2||M_2|) = 2\log(|M_2|) < |M_2|$. Hence, when $|M_1| > 0$, we have $\log(|M_1||M_2|) < |M_2|$. Therefore, the preprocess of sorting improves the efficiency.

When some messages are removed from either M_1 or M_2, new pairs may become mutually nearest. If $m_1 \in M_1$ and $m_2 \in M_2$ are mutually nearest, then they remain so even if some other messages are removed from either M_1 or M_2. By Proposition 5.3, for the purpose of finding the required pair in each iteration, it is sufficient to sort the list of pairs by increasing distance and traverse it once.

5.3. Computing the Mutually Nearest Matching

An algorithm to construct μ is presented in Figure 5. Initially, the algorithm creates an array of all pairs of $M_1 \times M_2$. The array is sorted by increasing distance. Instead of actually constructing M_1^* and M_2^*, the hash map H_1 counts how many occurrences of each message of M_1 are still eligible to be included in the constructed matching. The hash map H_2 does the same for messages of M_2. The ith iteration of the loop of line 15 checks (in line 19) whether the ith pair is eligible, and if so, line 20 adds that pair to the matching and lines 21 and 22 update H_1 and H_2, respectively.

The MND, denoted by $dist_{mn}$, is defined by means of the mutually nearest matching μ. It is the average distances over all pairs of μ.

Definition 5.4 (*Mutually Nearest Distance*). Given two sets of messages M_1 and M_2, the *mutually nearest distance* is the average of $dist(m_1, m_2)$ over all pairs (m_1, m_2) of the mutually nearest matching μ of M_1 and M_2.

Note that if M_1 and M_2 are not of equal size, then μ has $|M_1||M_2|$ pairs.

Mutually-Nearest Matching (M_1, M_2)

Input: Two sets of messages M_1, M_2
Output: A matching μ.

```
1: μ ← ∅
2: if |M1| > |M2| then
3:    swap M1 and M2 so that M1 will be the smaller set
4: create a one-dimensional array A of length |M1| · |M2|
5: create a hash map H1 of size |M1|
6: create a hash map H2 of size |M2|
7: for each m1 ∈ M1 and m2 ∈ M2 do
8:    add (m1, m2, dist(m1, m2)) to A
9: for each m1 ∈ M1 do
10:   add (m1, |M2|) to H1 with m1 as the search key and |M2| as the value
11: for each m2 ∈ M2 do
12:   add (m2, |M1|) to H2 with m2 as the search key and |M1| as the value
13: sort A in ascending order of the distances
14: i ← 1
15: while |μ| < |M1| · |M2| do
16:   let (m1, m2, dist(m1, m2)) be the ith element in A
17:   i ← i + 1
18:   let x = min{H1(m1), H2(m2)}
19:   if x ≠ 0 then
20:      add x copies of (m1, m2) to μ
21:      H1(m1) ← H1(m1) − x
22:      H2(m2) ← H2(m2) − x
23: return μ
```

Fig. 5. Computing the mutually nearest matching μ of two sets M_1 and M_2.

5.4. On the Relationship Between MND and EMD

We now compare the MND to the EMD, which was presented in Section 4.3. Although the computation of MND is much more efficient than that of EMD, frequently they provide similar results. To see this, consider two sets M_1 and M_2, and let c_2 be the center of mass of M_2. For simplicity, we assume that both sets have the number n of messages. Let μ_{mn} and μ_{em} be the matchings created in the computation of MND and EMD, respectively. Then for each $(m_1, m_2) \in \mu_{mn}$, there exists a pair $(m_1, m_2') \in \mu_{em}$, because μ_{em} is a matching. Based on the triangle inequality,

$$dist(m_1, m_2) \leq dist(m_1, m_2') + dist(m_2', c_2) + dist(c_2, m_2). \tag{1}$$

By a slight abuse of notation, we consider the MND matching μ_{mn} and the EMD matching μ_{em} as two-way functions that for a given m (in either M_1 or M_2) return its matching message in the other set. Thus, m_2 is $\mu_{mn}(\mu_{em}(m_2'))$. By Equation (1),

$$\begin{aligned} n \cdot dist_{mn}(M_1, M_2) &= \sum_{(m_1, m_2) \in \mu_{mn}} dist(m_1, m_2) \\ &\leq \sum_{(m_1, m_2') \in \mu_{em}} [dist(m_1, m_2') + dist(m_2', c_2) + dist(c_2, \mu_{mn}(\mu_{em}(m_2')))]. \end{aligned}$$

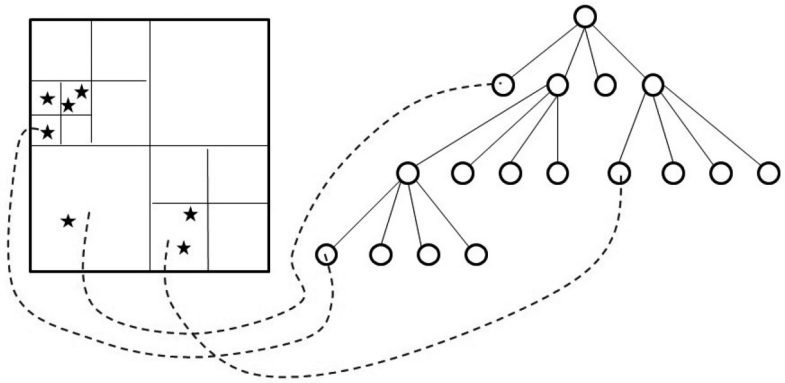

Fig. 6. A quad-tree index (on the right) and the area it partitions, given a threshold of 2. The dashed lines illustrate the correspondence between the nodes of the tree and the rectangular areas.

By definition, $\sum_{(m_1,m_2')\in\mu_{em}} dist(m_1, m_2') = n \cdot dist_{em}(M_1, M_2)$. Let d_{c_2} be the average distance from messages of M_2 to c_2 (which is the center of mass of M_2)—that is, $d_{c_2} = (\sum_{m\in M_2} dist(m, c_2))/n$. Then $\sum_{(m_1,m_2')\in\mu_{em}} dist(m_2', c_2) = n \cdot d_{c_2}$. Similarly, $\sum_{(m_1,m_2')\in\mu_{em}} dist(c_2, \mu_{mn}(\mu_{em}(m_2'))) = n \cdot d_{c_2}$. It follows that $n \cdot dist_{mn}(M_1, M_2) \leq n \cdot dist_{em}(M_1, M_2) + 2n \cdot d_{c_2}$ (recall that $dist_{em}$ denotes the EMD). Consequently,

$$dist_{em}(M_1, M_2) \leq dist_{mn}(M_1, M_2) \leq dist_{em}(M_1, M_2) + 2d_{c_2}.$$

In other words, $|dist_{mn}(M_1, M_2) - dist_{em}(M_1, M_2)| \leq 2d_{c_2}$. Thus, when d_{c_2} is small (which happens frequently when considering messages of real users), the two distance measures provide similar results, but differently from EMD, MND can be computed efficiently.

6. QUAD-TREE DISTANCE

In this section, we present QTD, which is another new measure. We show how to compute it and discuss its advantages.

A disadvantage of the previous methods, especially those that are based on computing a matching, is that they cannot cope efficiently with changes to the sets of messages. In particular, consider two sets M_1 and M_2 such that the distance d between them has already been computed. An important question is how to incrementally compute the distance between M_1 and $M_2 \cup \{m\}$ (where m is a new message added to M_2) from the previous value d, and similarly for M_1 and $M_2 \setminus \{m\}$. Next, we present a distance measure that uses a quad-tree index and is highly efficient, even when the sets are updated frequently.

Quad trees are index structures for sets of points on the plane [Samet 1984, 2005]. A quad-tree index is constructed by recursively partitioning a rectangular area into four equal parts. The four created rectangles are the *children* of the partitioned one. Each point is associated with its immediately containing rectangle. The stopping condition of the recursion is reached when the number of points in the rectangular area is below a given threshold. For example, Figure 6 presents a quad tree that was created with a threshold of 2 (i.e., every rectangle that contains more than two points is partitioned). The tree index is constructed by creating a node from each rectangle. The root is the initial rectangle. For each rectangle, its children are the four parts into which it was divided.

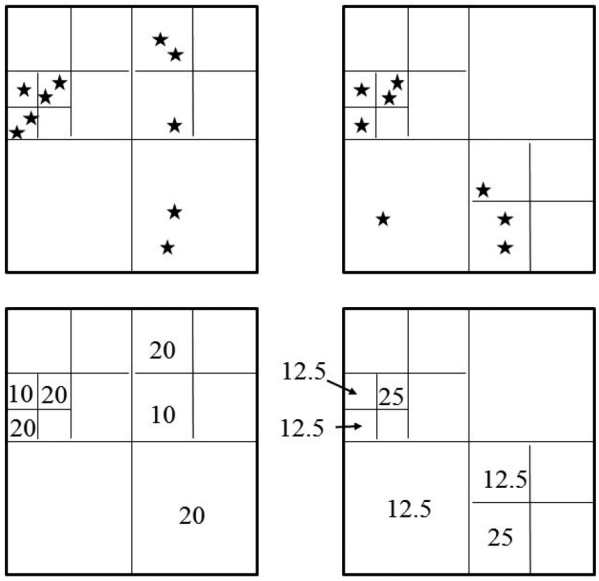

Fig. 7. Two quad trees (on the top) and the spatial distributions they yield (on the bottom). The numbers are the masses of the leaves.

A node is a *leaf* if it has no children; otherwise, it is an *internal node*. The *depth* of a node v, denoted by $depth(v)$, is the length of the path from the root of the tree to v—namely, it is 0 for the root itself, 1 for the children of the root, and so forth. The depth of a tree is the maximum over the depths of its nodes. Due to efficiency considerations, our implementation arbitrarily bounds the depth of the tree—that is, we do not divide nodes with a depth that is equal to the bound.

Two quad trees are *corresponding* if their roots refer to rectangular areas with the same coordinates. In other words, the upper left corners of the two rectangles have the same coordinates, and so do the lower right corners. The correspondence relation is extended from the roots to other nodes of the two trees as follows. Two nodes (one from each tree) are *corresponding* if they represent the same rectangular area of the plane.

The common use of a quad tree is as an index structure. We, however, use quad trees to define a distance measure for point sets. A quad tree can be used for outlining the spatial distribution of the points on the plane. To that end, we define the *mass* of a node as the percentage of the points that are inside its rectangular area. In Figure 7, for example, the left tree has 10 points in total, so a node that contains 2 points has a mass of 20%. The right tree indexes 8 points, so a node that contains 2 points has a mass of 25%.

Consider two corresponding quad trees. We compare the distributions of their sets of points by means of the masses of their nodes. Since the effect of each point should not be considered more than once, we only compare a pair of nodes if they are corresponding and (at least) one of them is a leaf. We say that such a pair of nodes is *comparable*. For every two nodes that are comparable, we find the difference in their masses and multiply it by the width of the rectangle they represent. The sum of the results over all comparable pairs, denoted by $dist_{\mathrm{qt}}$, is called the *quad-tree distance*.

Example 6.1. Consider the trees in Figure 7. Suppose that the width of the root is 1m, each child of the root has a width of 0.5m, and a child of a child has a width of $(\frac{1}{2})^2 =$ 0.25m. Thus, the upper left quarter yields $(|12.5 - 10| + |25 - 20| + |12.5 - 20|) * 0.25$.

Quad-Tree Distance (T_1, T_2)

Input: Two quad trees T_1, T_2 created for sets of messages M_1, M_2
Output: The quad-tree distance between the trees.

1: **return** Node-Distance ($T_1.root$, $T_2.root$)

Node-Distance (v_1, v_2)

Input: Two nodes v_1, v_2
Output: A distance d

1: $d \leftarrow 0$
2: **if** $isLeaf(v_1)$ or $isLeaf(v_2)$ **then**
3: $\quad d \leftarrow d + (c^{\,depth(v_1)} \cdot width(v_1)) \cdot (|v_1.mass - v_2.mass|)$
4: **else**
5: $\quad$ **for** $i = 1$ to 4 **do**
6: $\quad\quad d \leftarrow d$+*Node-Distance* $(v_1.child[i], v_2.child[i])$
7: **return** d

Fig. 8. Recursively computing the QTD for two corresponding nodes.

The upper right quarter contributes $|(10+20)-0| * 0.5$, and the lower left quarter contributes $|12.5-0| * 0.5$ (in these two cases, a comparable pair comprises children of the root and has a width of 0.5m). The lower right quarter provides $|(25 + 12.5) - 20| * 0.5$. Hence, the distance is $(|12.5-10| + |25-20| + |12.5-20|) * 0.25 + |30-0| * 0.5 + |12.5-0| * 0.5 + |37.5-20| * 0.5 = 33.75$.

To intuitively understand the notion of QTD, consider two clusters of messages—one in M_1 and the other in M_2—such that both are in the same area and have proportionally similar sizes. In this case, the masses of the corresponding nodes are similar and hence their contribution to $dist_{\mathrm{qt}}$ is small. On the other hand, if some area M_1 has many messages whereas M_2 has only a few, then the difference in the masses will contribute significantly to $dist_{\mathrm{qt}}$.

QTD is effective in measuring spatial similarity because it takes into account the distribution of the messages in every part of the considered area.

In a city, users move on roads and hence the actual travel distance from one point to another is not Euclidean (or Haversine). The difference between the Euclidean and the actual travel distances is called *circuity* [Ballou et al. 2002]. It decreases exponentially as a function of the Euclidean distance between the points. Thus, for near points, the expected circuity is larger than for distant ones. To adjust the QTD to this phenomenon, we assume that the width decreases from a parent to a child by a factor of $\frac{c}{2}$ (rather than the actual $\frac{1}{2}$), where c is the *circuity factor*. In our experiments, we used $c = 1.6$ because it gave the best results.

We compute QTDs as follows. Consider two sets of messages M_1 and M_2, such that all of their locations are contained in a bounding rectangle B. Suppose that T_1 and T_2 are the quad trees of M_1 and M_2, respectively, both with a root corresponding to B. The computation of the distance is by calling *Node-Distance*($T_1.root$, $T_2.root$), which is the recursive method of Figure 8.

We maintain a quad tree for the messages of each user. Since a quad tree is an index, updates are done efficiently. The quad tree also keeps the following information. Each leaf v has the number $n(v)$ of its messages. The quad tree stores the total number $|M|$ of

messages. Thus, when inserting or deleting a message, only M and $n(v)$ of the relevant leaf have to be changed. The mass of a leaf v is $n(v)/|M|$. For an internal node, the number of its messages is the sum of $n(v)$ over all of its descendant leaves v. Thus, the method of Figure 8 can be efficiently implemented.

Two parameters control the efficiency and effectiveness of our method. The *depth bound* determines the maximum number of levels in the tree. The *partition threshold* sets the minimal number of points (messages) in a leaf that triggers a partition. These two parameters affect the size of the tree and the number of messages in nodes. When the tree is too large, the computation of the distance will be inefficient. When the tree is too small, the QTD will be an imprecise measure of similarity, especially for users with a relatively few messages. In the tests of Section 8, we used a depth bound of 6 and a partition threshold of 3. These parameters provide a good balance between accuracy and efficiency.

7. FINDING THE *K*-MOST-SIMILAR USERS UNDER THE MUTUALLY NEAREST MEASURE

In this section, we assume that U is a set of users and u is another user. The goal is to compute the k users of U that are most similar to u. The naive approach is to calculate the distance between u and every user of U. However, as shown in the experiments of Section 8, this approach is costly for $dist_{mn}$ and $dist_{nn}$. Hence, we present optimization techniques for $dist_{mn}$ (which is the more accurate method among $dist_{mn}$ and $dist_{nn}$).

7.1. An Optimization Based on CMD

Our main technique is to use $dist^o_{cm}$, which is easily computed, to eliminate from consideration some users of U, when the distance measure is $dist_{mn}$. This optimization technique is based on the following proposition.

PROPOSITION 7.1. *Let M_1 and M_2 be two sets of messages. Then $dist^o_{cm}(M_1, M_2) \leq dist_{mn}(M_1, M_2)$.*

PROOF. Let $|M_1| = s$, $M_1 = \{m_{11}, \ldots, m_{1s}\}$, $|M_2| = t$, and $M_2 = \{m_{21}, \ldots, m_{2t}\}$. Recall that each message m of M_1 is duplicated t times in the set M^*_1; similarly, M^*_2 has s copies of each message of M_2. Moreover, all duplicates of a message m have the same location as m. Thus, M_1 and M^*_1 have the same center of mass, and similarly for M_2 and M^*_2. Let c_1 and c_2 be the centers of mass of M_1 and M_2, respectively.

Note that each one of M^*_1 and M^*_2 has $h = s \cdot t$ messages. We denote the messages of M^*_1 as m^*_{1i} and those of M^*_2 as m^*_{2i} $(1 \leq i \leq h)$. Recall that the mutually nearest matching μ of the sets M_1 and M_2 is actually a one-to-one correspondence between the messages of M^*_1 and M^*_2. We assume that the mutually nearest matching μ consists of the pairs (m^*_{1i}, m^*_{2i}) where $1 \leq i \leq h$ (note that we can always order the messages so that it will hold).

Next, we apply a linear transformation so that the line connecting c_1 and c_2 becomes the new x-axis. This transformation requires applying just rotation and translation. Hence, it does not change the distances between locations.

Since the centers of mass c_1 and c_2 are on the x-axis, we get that $\sum_{i=1}^{s} m_{1i}.y = 0$ and $\sum_{j=1}^{t} m_{2j}.y = 0$. Thus, the distance between c_1 and c_2 satisfies the following equations.

$$dist^o_{cm}(M_1, M_2) = \left| \frac{\sum_{j=1}^{t} m_{2j}.x}{t} - \frac{\sum_{i=1}^{s} m_{1i}.x}{s} \right| \tag{2}$$

$$= \left| \frac{\sum_{j=1}^{h} m^*_{2j}.x}{h} - \frac{\sum_{i=1}^{h} m^*_{1i}.x}{h} \right| \tag{3}$$

$$= \left| \frac{\sum_{j=1}^{h} (m_{2j}^*.x - m_{1j}^*.x)}{h} \right| \tag{4}$$

$$\leq \frac{\sum_{j=1}^{h} \left| m_{2j}^*.x - m_{1j}^*.x \right|}{h} \tag{5}$$

$$\leq \frac{\sum_{j=1}^{h} dist(m_{2j}^*, m_{1j}^*)}{h} \tag{6}$$

$$= dist_{mn}(M_1, M_2) \tag{7}$$

Equation (2) holds because the centers of mass are on the x-axis, so we only need to sum the x coordinates of the locations. Equation (3) follows from the fact that M_1 and M_1^* have the same center of mass and similarly for M_2 and M_2^*. Equations (4) and (5) are obvious. Since actual distances should also take the y coordinates into account, Equation (6) holds. Finally, Equation (7) holds because the pairs (m_{1i}^*, m_{2i}^*), for $1 \leq i \leq h$, constitute the mutually nearest matching μ. □

To efficiently find the k users of U that are most similar to u, we apply Proposition 7.1 as follows. Let M_u and $M_1, \ldots, M_p$ be the sets of messages of u and of the users of U, respectively. First, we compute the centers of mass of all of the sets. Next, we sort $M_1, \ldots, M_p$ according to the distances between their centers of mass and that of M_u. Let $S = M_{j_1}, \ldots, M_{j_p}$ be the resulting sorted list—that is, $dist_{cm}^o(M_u, M_{j_a}) \leq dist_{cm}^o(M_u, M_{j_b})$ for all $1 \leq a < b \leq p$.

To compute the k-most-similar users, we use a priority queue H of size k that is initialized to contain $M_{j_1}, \ldots, M_{j_k}$. The priority is determined by the MND $dist_{mn}$ from M_u; a longer distance means a higher priority. For $i > k$, sets are added to H according to their order in S. When adding M_{j_i}, we first compute its MND $dist_{mn}$ to the set M_f at the top of H.

If $dist_{mn}(M_f, M_u) \leq dist_{cm}^o(M_{j_i}, M_u)$, then H contains the k-most-similar users. In proof, S is sorted and so $dist_{cm}^o(M_{j_i}, M_u) \leq dist_{cm}^o(M_{j_{i'}}, M_u)$ for all $i' \geq i$. According to Proposition 7.1, $dist_{cm}^o(M_{j_{i'}}, M_u) \leq dist_{mn}(M_{j_{i'}}, M_u)$. Hence, $dist_{mn}(M_f, M_u) \leq dist_{mn}(M_{j_{i'}}, M_u)$ for all $i' \geq i$. The set M_f of H has the longest MND to M_u (among the sets of H). Hence, the claim is true and we stop.

If $dist_{mn}(M_f, M_u) > dist_{cm}^o(M_{j_i}, M_u)$, then first we compute $dist_{mn}(M_{j_i}, M_u)$. Then we test whether $dist_{mn}(M_f, M_u) > dist_{mn}(M_{j_i}, M_u)$. If so, we insert M_{j_i} into H and remove M_f. The code of this algorithm is presented in Figure 9.

We now analyze the complexity of the algorithm. Under the assumption that the number of messages of each user is bounded, the computation of either $dist_{mn}$ or $dist_{cm}^o$ takes $O(1)$ time. Hence, the time complexity of the algorithm is $O(|U| \log |U|)$ because of the sort of U in line 7. Note that other than the sort, the time complexity of the rest of the algorithm is linear in the size of U. If the maximal number x of messages of a user is unbounded, then the computation of $dist_{cm}^o$ is linear in x and that of $dist_{mn}$ takes $O(x^2 \log x)$ time. The distances with respect to u are computed for each user of U—that is, for $|U|$ users. Thus, the overall time complexity is $O(|U|log|U| + |U|x^2 \log x)$.

7.2. Early Termination Optimization

Recall that the algorithm of Figure 5 is needed for computing the MND between two sets. We modify this algorithm so that in each iteration of line 15, it computes the distance d according to the matching found thus far. In other words, at the end of each iteration, we get $d = \sum_{(m_1, m_2) \in \mu} dist(m_1, m_2)$. It thus follows that $\frac{d}{i}$ is a lower bound on

k-Most Similar Users (u, U)

Input: A user u, a set U of users
Output: The k users of U that are the most similar to u, according to their messages and $dist_{mn}$.

```
1: let M_u be the set of messages of u
2: let S be an array of size |U|
3: for i = 1 to |U| do
4:    let M_i be the set of messages of u_i
5:    d_i ← dist^o_cm(M_i, M_u)
6:    add (M_i, u_i, d_i) to S[i]
7: sort S by the distances d_i
8: let H be an empty priority heap
9: for i = 1 to k do
10:   let (M_j, u_j, d_j) be the element S[i]
11:   insert (M_j, u_j, dist_mn(M_j, M_u)) into H
12: stop_cond← false
13: i ← k + 1
14: while (not stop_cond) and (i ≤ |U|) do
15:   let (M_j, u_j, d_j) be the element S[i]
16:   let (M_f, u_f, d_f) be the element with the largest distance value, in H
17:   if dist_mn(M_f, M_u) ≤ d_j then
18:      stop_cond← true
19:   else
20:      if dist_mn(M_f, M_u) > dist_mn(M_j, M_u) then
21:         remove (M_f, u_f, d_f) from H
22:         insert (M_j, u_j, dist_mn(M_j, M_u)) into H
23:         i ← i + 1
24: let U_k be the set of users corresponding to the elements of H
25: return U_k
```

Fig. 9. Computing the k-most-similar users, for a given user u, according to $dist_{mn}$.

the MND at the end of the ith iteration—that is, $\frac{d}{i} \leq dist_{mn}(M_1, M_2)$, because pairs (m_1, m_2) are added to μ in the order of ascending distance between m_1 and m_2.

Our second optimization uses the preceding lower bound for an early-termination test when employing the algorithm of Figure 5 in the process of finding the k-most-similar users. As earlier, we use a priority queue H that stores the k-most-similar users found thus far. Initially, we insert into H the first k sets for which we have computed the MND to M_u. Suppose that $M_{u'}$ is the next set for which we compute $dist_{mn}(M_{u'}, M_u)$. We can terminate the computation as soon as $\frac{d}{i} > dist_{mn}(M_f, M_u)$, where M_f is the set at the top of H (i.e., farthest away from M_u). If this test does not cause an early termination and $dist_{mn}(M_f, M_u) > dist_{mn}(M_{u'}, M_u)$ at the end of the matching computation, then we insert $M_{u'}$ into H and remove M_f.

8. EXPERIMENTAL EVALUATION

We tested our methods on real-world data. The goals of our experiments were (1) to evaluate the effectiveness of the different measures in terms of the correlation between

the distance and user similarity, (2) to determine the efficiency of the distance measures by comparing the running times of finding the k-most-similar users, and (3) to check the effect of our optimization techniques on the running time of the computation of $dist_{mn}$.

8.1. Description of the Tests

We now describe the setting of the tests and our methodology.

8.1.1. Experimental Setting. Our experiments were conducted over geotagged messages that were posted on Twitter (tweets) from Los Angeles (LA), New York City (NYC), and London. Each one of the three datasets contained more than 200,000 messages of approximately 25,000 users. All experiments were performed on all three datasets, and in all cases the results were the same, with only negligible differences between the datasets. To avoid repetition, we only present the results for the LA dataset.

In the LA dataset, for users with 5 messages or more, the average number of messages per user is 10.5, with a standard deviation of 0.81. The number of unique locations from which messages are posted, per user, is 8.8, with a standard deviation of 0.87. For users with at least 10 messages, the average number of messages per user is 21.1, with a standard deviation of 0.02. The average number of unique places from which messages were posted, per user, is 17.15, with a standard deviation of 0.42. This shows that most of the users post most of their messages from various locations.

The experiments were conducted on a machine equipped with a Core i5-3230M 2.6GHz processor, 8GB RAM, and a Windows 8 64-bit operating system. The code is written in Java and was executed using a 2GB heap (i.e., the JVM was run with the parameters -Xms2048M -Xmx2048M).

8.1.2. Methodology. To evaluate the effectiveness of a distance measure (i.e., its correlation with similarity among users), we determine its accuracy, as defined in Section 3. To obtain the data for experimentally measuring accuracy, we apply the following idea of *splitting*. We assume that there is a given set U of distinct users, each one with her own set of messages. (Recall that each of our three datasets is such a U.) A user $u \in U$ is *split* by randomly dividing the set of messages M_u into two disjoint parts M_u^1 and M_u^2—that is, $M_u^1 \cap M_u^2 = \emptyset$ and $M_u^1 \cup M_u^2 = M_u$. We denote by u^1 and u^2 the *virtual users* whose sets of messages are M_u^1 and M_u^2, respectively. Since the splitting is done randomly, the two virtual users u^1 and u^2 should have the same characteristics. Therefore, we say that u^1 is the *matching* user of u^2 and vice versa. Let $\hat{U}$ be the set of all virtual users (i.e., those obtained by splitting each $u \in U$).

We measure accuracy as follows. For each user $u \in U$, we create the set $\hat{U}^u$ by removing u^2 from $\hat{U}$ (i.e., the matching user of u^1), whereas all other virtual users remain in $\hat{U}^u$. Now we test whether u^1 is one of the k-most-similar users of u^2 among those of $\hat{U}^u$. Recall that if the answer is positive, we call it a *success*. The rationale for considering it a success is that u^1 and u^2 have the same characteristics (these are messages of the same user) and thus should be deemed similar. We repeat this test for each $u \in U$. The percentage of successes (in the $|U|$ tests) is the measured accuracy. It should be emphasized that the splitting is done randomly, and therefore the measured accuracy is unbiased.

We also measure the running time as follows. For a given u^2, a *search* is the task of finding the k-most-similar users of u^2 among those of $\hat{U}^u$. We report the average running time of a search over all $u \in U$.

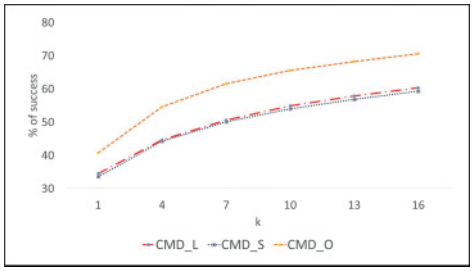

Fig. 10. Accuracy of the versions of CMD as a function of k.

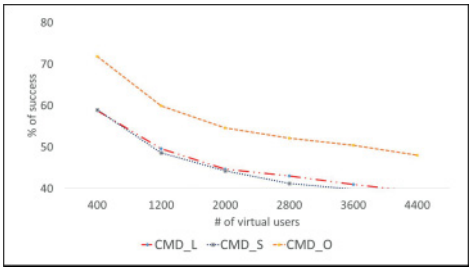

Fig. 11. Accuracy of the versions of CMD per the number of virtual users.

We now list the parameters involved in our experiments and their default values:

—The threshold τ is the minimal number of messages that a user must have before the splitting is done. The default is $\tau = 10$.
—The set U of users has a default size of 1,000, which after the split provides 2,000 virtual users. The users of U are uniformly chosen from the members of the LA dataset (Section 8.1.1) that satisfy the threshold τ.
—The number k of answers (i.e., the k-most-similar users). The default is $k = 4$.
—The portion p of messages in the larger subset when the splitting is done. The default is $p = 0.5$ (i.e., the splitting is into two subsets of equal size).

In the experiments, we measure accuracy and running times. We do it by varying one parameter while keeping all others at their default values.

8.2. Results

We now present the results of our experiments. First, in Section 8.2.1, we empirically compare the different variations of CMD and NND. Second, in Sections 8.2.2 and 8.2.3, we compare the effectiveness and the efficiency of all methods as a function of the threshold τ, the number of users $|U|$, the value k, and the portion p. Third, the effect of the optimization on MND is shown in Section 8.2.4. Finally, in Section 8.2.5, we present the running time of building the quad-tree index.

8.2.1. Testing Versions of Existing Methods. In Section 4, we presented three versions of the CMD and two versions of the NND. Our first set of tests compares the different versions of the methods to simplify the comparison of existing methods to the novel ones.

Figures 10 and 11 show the accuracy of the three versions of the CMD. We denote by CMD_L, CMD_S, and SMD_O the similarity detection using $dist^l_{cm}$, $dist^s_{cm}$, and $dist^o_{cm}$, respectively. In Figure 10, the accuracy is presented as a function of k. In Figure 11, the accuracy is measured as a function of the number of users in the initial set U. In both cases, CMD_O is more accurate than the other two versions of CMD. The reason for this is that CMD_O reduces the effect of outliers in comparison to the other methods.

Figures 12 and 13 present the running times (in milliseconds) of the three versions of CMD as a function of k and as a function of $|U|$, respectively. Clearly, CMD_O is more efficient than the other two versions, because it merely computes a distance between the two centers of mass rather than computing the distance for each point in one of the sets. These experiments clearly show that CMD_O is a better method than CMD_L and CMD_S, so in the following experiments we will only compare the new methods to CMD_O (and omit CMD_L and CMD_S).

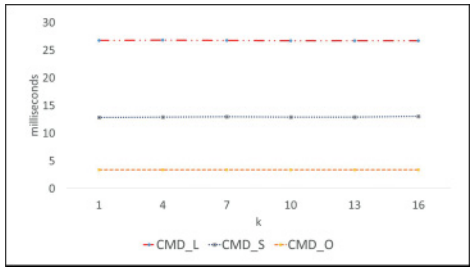

Fig. 12. Running times of the versions of CMD as a function of k.

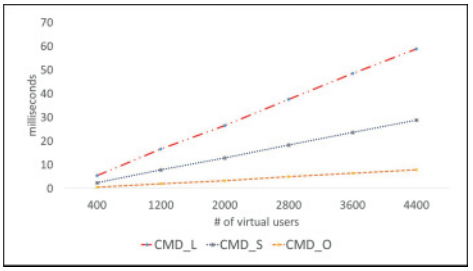

Fig. 13. Running times of the CMD versions per the number of virtual users.

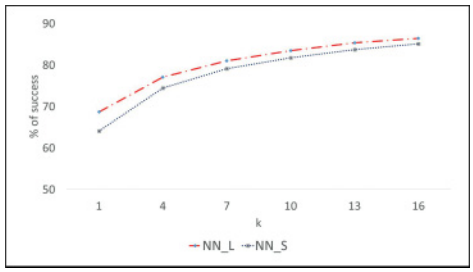

Fig. 14. Accuracy of the versions of NND as a function of k.

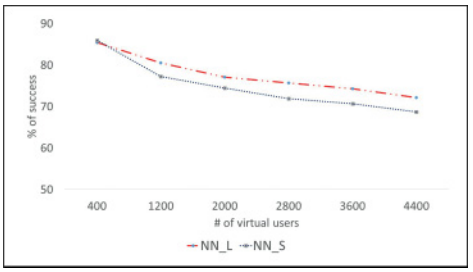

Fig. 15. Accuracy of the versions of NND per the number of virtual users.

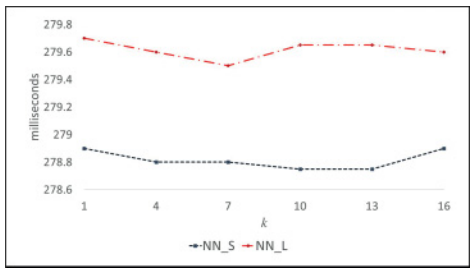

Fig. 16. Running times of the versions of NND as a function of k.

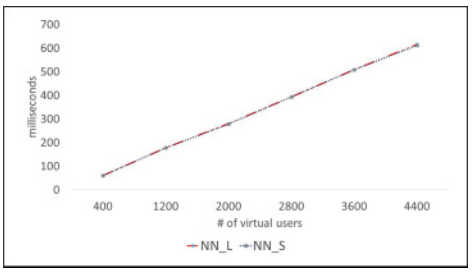

Fig. 17. Running times of the NND versions per the number of virtual users.

To compare the two versions of NND, we computed the accuracy and the running times of using $dist^l_{nn}$ (denoted by NND_L) and $dist^s_{nn}$ (denoted by NND_S) as a function of k and as a function of $|U|$. Figures 14 through 17 show that NND_L is slightly more accurate than NND_S, but it is also slightly less efficient than NND_S, so for NND no version completely outperforms the other one.

8.2.2. Effectiveness. We now present experiments with the new methods, including comparison to existing ones. Figure 18 presents the accuracy of the different methods as a function of k. Figure 19 presents the accuracy as a function of the number of virtual users. Figure 20 presents the accuracy for different partitions of the users. Clearly, CMD_O is the least accurate method. The HD also has low accuracy. The NND_L and NND_S methods are fair, but MND and the QTD outperform them. Clearly, QTD provides the highest accuracy among all methods we tested.

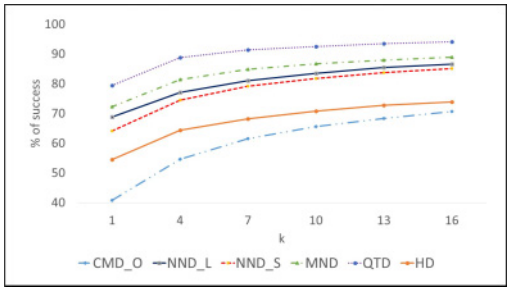

Fig. 18. Accuracy as a function of k.

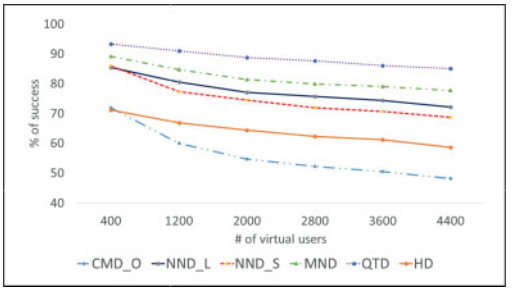

Fig. 19. Accuracy as a function of the number of virtual users.

Table II. Running Times, in Milliseconds, of QTD and CMD as a Function of k

k	1	4	7	10	13	16
QTD	1.00	1.00	0.99	1.05	1.00	1.00
CMD	3.57	3.61	3.38	3.49	3.46	3.46

Table III. Running Times, in Milliseconds, of QTD and SMD as a Function of the Number of Users

Number of users	400	1,200	2,000	2,800	3,600	4,400
QTD	0.18	0.48	0.99	1.64	2.35	3.07
CMD	0.65	2.02	3.53	4.83	6.38	7.78

Figure 20 is the first experiment in which each set of messages is partitioned into unequal parts. A portion of 0.7 means that for each user u, we partition M_u into two sets: one of size $0.3|M_u|$ and the other of size $0.7|M_u|$. Similarly, for a portion of 0.8, the partition is into two sets having sizes that are 20% and 80% of the original one. In this test, QTD failed to create an index for very small sets due to the parameters we used for the construction of the tree. For example, consider a user with 10 messages that is partitioned into a virtual user with 8 messages and a virtual user with 2 messages. The tree cannot have a leaf with just 2 messages, so the method fails. For a portion of 0.8, NND_S provides the best results because as the size of the smaller set decreases, it becomes easier to find a nearest neighbor in the larger part and the accuracy grows.

Figure 21 presents the accuracy as a function of the threshold τ (i.e., the minimal number of messages that a user can have). Obviously, when the threshold increases, it is easier to find the matching virtual user of a given one. However, we can see that HD is less affected than CMD_L when increasing the threshold, because the former is determined based on a single point. Consequently, when the number of messages per user is large, HD is the least effective method.

8.2.3. Efficiency. To determine the efficiency of the distance measures, we tested the running times in various cases. Figure 22 and Table II present the running times as a function of k. Each point on the graph (and each value presented in the table) is the average running time over 1,000 searches. Figure 23 and Table III present the running time as a function of the number of users. Figure 24 and Table IV present the running times as a function of the partition sizes. Figure 25 and Table V present the running times as a function of the threshold. In all cases, HD is the least efficient and the least scalable measure. The two versions of NND are also quite inefficient. MND

Table IV. Running Times, in Milliseconds, of QTD and CMD for Different Partition Sizes

Portion	0.5	0.7	0.8
QTD	0.97	0.95	0.90
CMD	3.61	3.46	3.40

Table V. Running Times, in Milliseconds, of QTD and CMD as a Function of the Threshold

Threshold	8	12	18	25	35	50
QTD	0.80	0.84	0.92	0.91	0.94	0.99
CMD	3.64	3.18	3.79	4.42	5.09	6.18

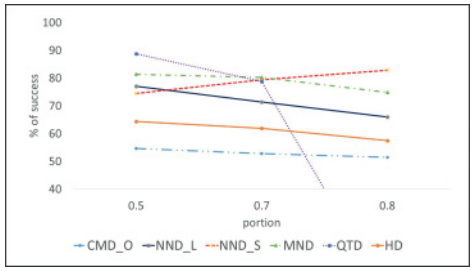

Fig. 20. Accuracy as a function of the partition of users into virtual users.

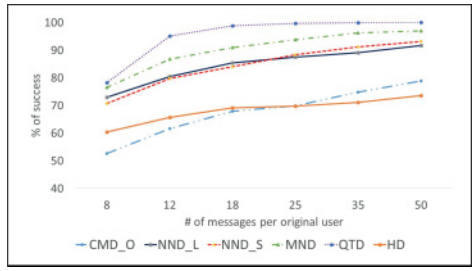

Fig. 21. Accuracy as a function of the threshold.

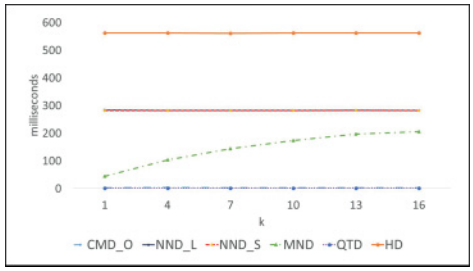

Fig. 22. Running times as a function of k.

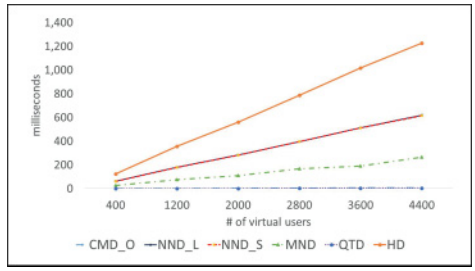

Fig. 23. Running times as a function of the number of virtual users.

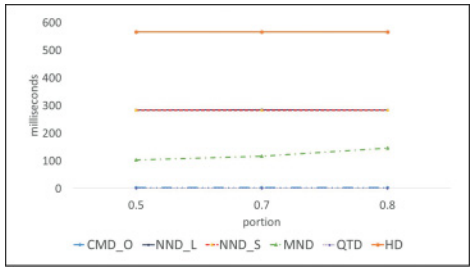

Fig. 24. Running times as a function of the partition sizes.

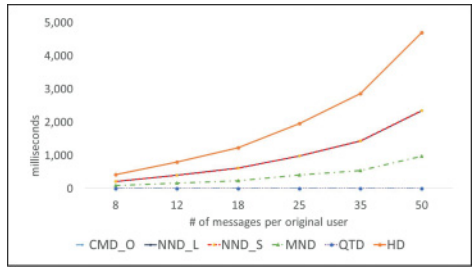

Fig. 25. Running times as a function of the threshold.

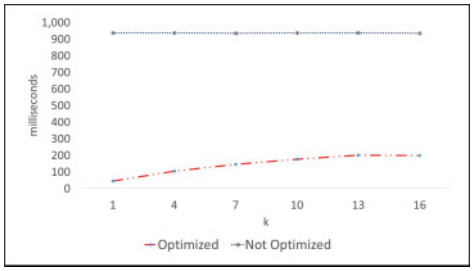

Fig. 26. Running times of MND, with and without optimizations, as a function of k.

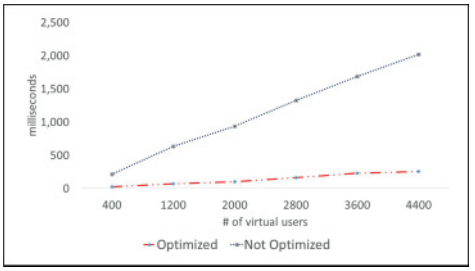

Fig. 27. Running times of MND, with and without optimizations, as a function of the number of virtual users.

Table VI. Build Time, in Milliseconds, of the Quad-Tree Index as a Function of the Number of Messages

No. of messages	25	50	100	500
Time (milliseconds)	2	3	5	20

with the proposed optimizations is better than NND and has acceptable running times. The methods CMD_O and QTD are highly efficient and scalable. However, the running times of QTD are for the case where the index is already built. In Section 8.2.5, we discuss the time needed for the index construction.

8.2.4. The Effect of the Optimizations of MND. In Section 7, we developed optimizations for MND. Their effect on the running times as a function of k and as a function of the number of users are presented in Figures 26 and 27, respectively. It can be seen that the optimization techniques significantly reduce the running time and improve the scalability of MND.

8.2.5. Building the Quad-Tree Index. The QTD is effective and efficient. However, if the index does not exist, it should be built. Recall that a separate quad tree should be built for each user, and the time it takes depends on the number of messages that the user has. Table VI presents the time needed for constructing a quad tree as a function of the number of messages. (Note that the parameters of Section 8.1.2 are irrelevant in this experiment.)

Building the index for thousands of users can be done in a few seconds, yet if the index does not exist, using MND, with the optimizations, is likely to be more efficient than building the index (i.e., compare the times of Table VI to those of Figure 27).

8.2.6. Validity of the Methodology. One conclusion from our experiments is that the places of user activities, in online geosocial networking sites, are not arbitrary. In other words, if the locations in our dataset were scattered uniformly over the area without any statistical significance, none of our methods could have distinguished between different users merely according to their messages. The ability of our methods to distinguish between users, with high accuracy, shows that user activities have unique patterns. Hence, using the locations of messages to detect similarity among users is a valid and useful approach. Figure 28 provides examples of messages of users to visualize this.

To show this formally, we use the p-value—a common probability measure in statistical hypothesis tests [Lehmann and Romano 2005]. In our validation, the null hypothesis that we try to reject is the following: for a user u chosen arbitrarily, all users have the same probability of being in the set of the k-most-similar users of u. The p-value is the probability of obtaining test results that are at least as high as the actual results,

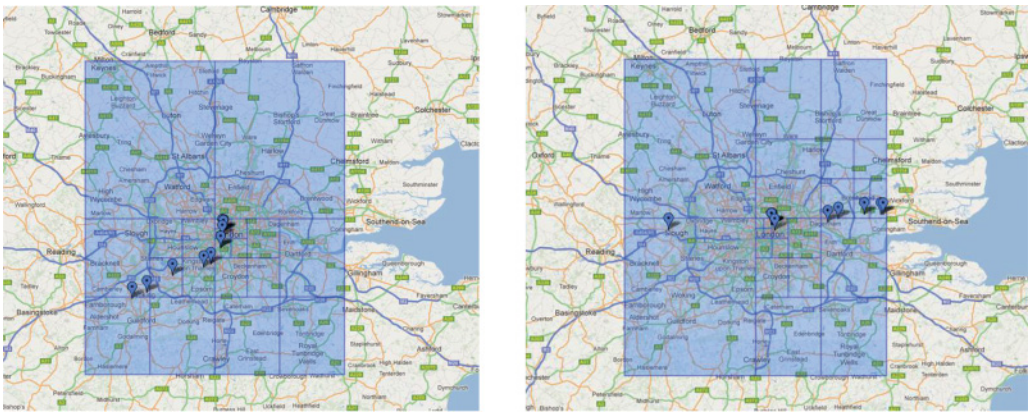

Fig. 28. Messages of real users and the quad trees that were built for them.

assuming that the null hypothesis holds. Typically, when the p-value is less than 0.05, the null hypothesis is deemed highly unlikely.

Suppose that we partition a user u into u_1 and u_2, and compare u_2 to N users. According to the null hypothesis, the probability that u_1 is in the k-most-similar users of u_2 is k/N. Since tests are independent, the probability of success in ℓ tests is $(k/N)^\ell$.

Suppose that we conduct 1,000 independent tests using a distance measure that has a 60% success rate (e.g., CMD_O), where $N = 1{,}000$ and $k = 10$. In this case, we would observe success in 600 out of the 1,000 tests. What is the probability of having 600 or more successes under the null hypothesis? The p-value in this case is

$$\sum_{i=600}^{1000} \left(\frac{k}{N}\right)^i \left(1 - \frac{k}{N}\right)^{(1000-i)} = \sum_{i=600}^{1000} (0.01)^i (0.99)^{(1000-i)} \ll 0.05.$$

This clearly rejects the null hypothesis and shows that geosocial similarity is statistically significant. For methods that are more accurate than CMD_O, such as NND_L, MND, and QTD, the p-value is even lower and the statistical significance is higher.

9. CONCLUSIONS

This work studies the problem of finding similarity among users based on the locations of their activities (e.g., posting messages) in social media. We discussed common applications that require effective methods for detecting geosocial similarity, such as sociospatial analysis, recommendation systems, identity detection, and privacy protection. We formulated geosocial similarity in terms of measuring the distance between sets of points. We discussed four existing measures, namely Hausdorff, three variants of center of mass, two variants of nearest neighbor, and EMD. We explained why these measures (except for the last one) cannot accurately detect geosocial similarity. As for EMD, it is accurate but not efficient. These conclusions are also supported by our experiments that are summarized later in this section.

We introduced two novel measures that are designed to handle geosocial similarity accurately and efficiently. The first is MND, which is suitable for ad hoc tasks. When sets have different sizes, it is not obvious how to apply this distance. We showed how to define and efficiently compute it in the general case. Furthermore, we developed optimization techniques for finding the k-most-similar users (to a given one) according to this distance measure. Our experiments show the importance of these optimizations,

which yield a speedup of up to 10-fold. We proved that the mutually nearest measure is a good approximation of the EMD (even though it is much more efficient).

Although a quad tree is a common index structure, using it as a distance measure between sets is a novel idea. This measure is highly accurate and very efficient. It incurs the additional cost of building the indexes, but in many applications they may already exist, and if so, this is the preferred method.

To compare the distance measures, we introduced a novel test methodology and confirmed its validity in terms of its p-value. We believe that this methodology could be useful also in the context of other research problems and hence is an important contribution in and of itself. Using this methodology and Twitter data, we tested the accuracy and running time (of finding the k-most-similar users) for each distance measure.

Our experiments clearly indicate that the HD and CMD measures are inaccurate. In some experiments, their accuracy was less than 70% compared to 90% for the MND and QTD measures. For users with at least 50 messages, the accuracy of HD and CMD was about 70% and 75%, respectively, compared to 90% and 95% for the MND and QTD measures, respectively.

As for efficiency, for users with at least 50 messages, the running times of the CMD and QTD measures were instantaneous—that is, a few milliseconds. For the MND and the HD measures, the running times were approximately 1 and 5 seconds, respectively. Thus, the quad-tree and mutually nearest measures are superior to HD in terms of both accuracy and efficiency. Moreover, as the number of messages per user increases, this superiority grows. This is an important discovery due to the high popularity of the HD for comparing sets of points.

We showed that the two versions of the NND are only moderately accurate and efficient. They are outperformed, in terms of both accuracy and efficiency, by the MND measure. For example, for users with at least 50 messages, the accuracy of (both versions of) nearest neighbor was about 80% compared to 90% for the MND. The running times were approximately 1 and 2.5 seconds for MND and for (both versions of) the NND measure, respectively. These results are important, because the NND is frequently used in various tasks and is supported by many systems.

Future work includes generalizing our measures to take into account geographic properties of visited entities, such as their types, the semantics of locations, and the posting times, as well as combining locations with geographically related textual content of messages. Another topic for future work is how to use the edges of the social network (i.e., the social relationships between users) to generalize the measures. Note that taking into account the semantics of locations, the posting times, the textual content of messages, and the edges of the social network when computing the distance measures may significantly increase the running time and would limit the usage to datasets that can provide these features. An interesting question is whether existing measures (e.g., HD and NND) can be modified so that they could accurately serve as methods for geosocial similarity.

REFERENCES

Marco D. Adelfio, Sarana Nutanong, and Hanan Samet. 2011. Similarity search on a large collection of point sets. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS'11)*. ACM, New York, NY, 132–141. DOI: http://dx.doi.org/10.1145/2093973.2093992

Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 6, 734–749. DOI: http://dx.doi.org/10.1109/TKDE.2005.99

Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, NJ.

Deepa Anand and Kamal K. Bharadwaj. 2010. Enhancing accuracy of recommender system through adaptive similarity measures based on hybrid features. In *Proceedings of the 2nd International Conference on Intelligent Information and Database Systems: Part II (ACIIDS'10)*. 1–10. http://dl.acm.org/citation.cfm?id=1894808.1894810

Lars Arge, Octavian Procopiuc, Sridhar Ramaswamy, Torsten Suel, Jan Vahrenhold, and Jeffrey Scott Vitter. 2000. A unified approach for indexed and non-indexed spatial joins. In *Proceedings of the 7th International Conference on Extending Database Technology: Advances in Database Technology (EDBT'00)*. 413–429. http://dl.acm.org/citation.cfm?id=645339.650131

Lars Arge, Octavian Procopiuc, Sridhar Ramaswamy, Torsten Suel, and Jeffrey Scott Vitter. 1998. Scalable sweeping-based spatial join. In *Proceedings of the 24rd International Conference on Very Large Data Bases (VLDB'98)*. 570–581. http://dl.acm.org/citation.cfm?id=645924.671340

Jaime Ballesteros, Ariel Cary, and Naphtali Rishe. 2011. SpSJoin: Parallel spatial similarity joins. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS'11)*. ACM, New York, NY, 481–484. DOI:http://dx.doi.org/10.1145/2093973.2094054

Ronald H. Ballou, Handoko Rahardja, and Noriaki Sakai. 2002. Selected country circuity factors for road travel distance estimation. *Transportation Research Part A: Policy and Practice* 36, 9, 843–848.

Jie Bao, Yu Zheng, and Mohamed F. Mokbel. 2012. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, New York, NY, 199–208. DOI:http://dx.doi.org/10.1145/2424321.2424348

Catriel Beeri, Yaron Kanza, Eliyahu Safra, and Yehoshua Sagiv. 2004. Object fusion in geographic information systems. In *Proceedings of the 30th International Conference on Very Large Data Bases, Volume 30 (VLDB'04)*. 816–827. http://dl.acm.org/citation.cfm?id=1316689.1316760

Panagiotis Bouros, Shen Ge, and Nikos Mamoulis. 2012. Spatio-textual similarity joins. *Proceedings of the VLDB Endowment* 6, 1, 1–12. DOI:http://dx.doi.org/10.14778/2428536.2428537

Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. 2011. Structural trend analysis for online social networks. *Proceedings of the VLDB Endowment* 4, 10, 646–656. DOI:http://dx.doi.org/10.14778/2021017.2021022

Laurent Candillier, Frank Meyer, and Françoise Fessant. 2008. Designing specific weighted similarity measures to improve collaborative filtering systems. In *Proceedings of the 8th Industrial Conference on Advances in Data Mining: Medical Applications, E-Commerce, Marketing, and Theoretical Aspects (ICDM'08)*. 242–255. DOI:http://dx.doi.org/10.1007/978-3-540-70720-2_19

Bin Cao, Jian-Tao Sun, Jianmin Wu, Qiang Yang, and Zheng Chen. 2008. Learning bidirectional similarity for collaborative filtering. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases, Part I (ECML PKDD'08)*. 178–194. DOI:http://dx.doi.org/10.1007/978-3-540-87479-9_30

Bogdan Carbunar and Radu Sion. 2011. Private geosocial networking. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS'11)*. ACM, New York, NY, 365–368. DOI:http://dx.doi.org/10.1145/2093973.2094024

Bogdan Carbunar, Radu Sion, Rahul Potharaju, and Moussa Ehsan. 2012. The shy mayor: Private badges in geosocial networks. In *Proceedings of the 10th International Conference on Applied Cryptography and Network Security (ACNS'12)*. 436–454. DOI:http://dx.doi.org/10.1007/978-3-642-31284-7_26

Francesca Carmagnola, Francesco Osborne, and Ilaria Torre. 2014. Escaping the big brother: An empirical study on factors influencing identification and information leakage on the Web. *Journal of Information Science* 40, 2, 180–197. DOI:http://dx.doi.org/10.1177/0165551513509564

Peter J. Carrington, John Scott, and Stanley Wasserman. 2005. *Models and Methods in Social Network Analysis*. Cambridge University Press, New York, NY.

Yen-Yu Chen, Torsten Suel, and Alexander Markowetz. 2006. Efficient query processing in geographic Web search engines. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (SIGMOD'06)*. ACM, New York, NY, 277–288. DOI:http://dx.doi.org/10.1145/1142473.1142505

Kwok-Wai Cheung and Lily F. Tian. 2004. Learning user similarity and rating style for collaborative recommendation. *Information Retrieval* 7, 3–4, 395–410. DOI:http://dx.doi.org/10.1023/B:INRT.0000011212.66249.b7

Maria Christoforaki, Jinru He, Constantinos Dimopoulos, Alexander Markowetz, and Torsten Suel. 2011. Text vs. space: Efficient geo-search query processing. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, 423–432.

Cheng Ta Chung, Chia Jui Lin, Chih Hung Lin, and Pu Jen Cheng. 2014. Person identification between different online social networks. In *Proceedings of the 2014 IEEE/WIC/ACM International Joint Con-*

ferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Volume 01 (WI-IAT'14). IEEE, Los Alamitos, CA, 94–101. DOI: http://dx.doi.org/10.1109/WI-IAT.2014.21

Ian De Felipe, Vagelis Hristidis, and Naphtali Rishe. 2008. Keyword search on spatial databases. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering (ICDE'08)*. IEEE, Los Alamitos, CA, 656–665. DOI: http://dx.doi.org/10.1109/ICDE.2008.4497474

Patrick Doreian, Vladimir Batagelj, and Anuska Ferligoj. 2005. *Generalized Blockmodeling*. Cambridge University Press, New York, NY.

Yerach Doytsher, Ben Galon, and Yaron Kanza. 2011. Storing routes in socio-spatial networks and supporting social-based route recommendation. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks*. ACM, New York, NY, 49–56. DOI: http://dx.doi.org/10.1145/2063212.2063219

M. P. Dubuisson and A. K. Jain. 1994. A modified Hausdorff distance for object matching. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 1—Conference A: Computer Vision and Image Processing*. IEEE, Los Alamitos, CA, 566–568.

Thomas Eiter and Heikki Mannila. 1997. Distance measures for point sets and their computation. *Acta Informatica* 34, 103–133.

Kahina Gani, Hakim Hacid, and Ryan Skraba. 2012. Towards multiple identity detection in social networks. In *Proceedings of the 21st International Conference Companion on World Wide Web (WWW'12 Companion)*. ACM, New York, NY, 503–504. DOI: http://dx.doi.org/10.1145/2187980.2188098

Jennifer Golbeck. 2009. Trust and nuanced profile similarity in online social networks. *ACM Transactions on the Web* 3, 4, Article No. 12. DOI: http://dx.doi.org/10.1145/1594173.1594174

Philippe Golle and Kurt Partridge. 2009. On the anonymity of home/work location pairs. In *Proceedings of the 7th International Conference on Pervasive Computing (Pervasive'09)*. 390–397. DOI: http://dx.doi.org/10.1007/978-3-642-01516-8_26

Irena Grabovitch-Zuyev, Yaron Kanza, Elad Kravi, and Barak Pat. 2007. On the correlation between textual content and geospatial locations in microblogs. In *Proceedings of Workshop on Managing and Mining Enriched Geo-Spatial Data (GeoRich'14)*. ACM, New York, NY, Article No. 3. DOI: http://dx.doi.org/10.1145/2619112.2619115

Krzysztof Janowicz, Martin Raubal, Angela Schwering, and Werner Kuhn. 2008. Semantic similarity measurement and geospatial applications. *Transactions in GIS* 12, 6, 651–659. DOI: http://dx.doi.org/10.1111/j.1467-9671.2008.01129.x

Lei Jin, Hassan Takabi, and James B. D. Joshi. 2011. Towards active detection of identity clone attacks on online social networks. In *Proceedings of the 1st ACM Conference on Data and Application Security and Privacy (CODASPY'11)*. ACM, New York, NY, 27–38. DOI: http://dx.doi.org/10.1145/1943513.1943520

Yaron Kanza. 2016. Uncertainty in geosocial data: Friend or foe? *SIGSPATIAL Special* 8, 2, 3–10. DOI: http://dx.doi.org/10.1145/3024087.3024088

Yaron Kanza, Elad Kravi, and Uri Motchan. 2014. City nexus: Discovering pairs of jointly-visited locations based on geo-tagged posts in social networks. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'14)*. ACM, New York, NY, 597–600. DOI: http://dx.doi.org/10.1145/2666310.2666378

Xiangnan Kong, Jiawei Zhang, and Philip S. Yu. 2013. Inferring anchor links across multiple heterogeneous social networks. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM'13)*. ACM, New York, NY, 179–188. DOI: http://dx.doi.org/10.1145/2505515.2505531

Balachander Krishnamurthy and Craig E. Wills. 2009. On the leakage of personally identifiable information via online social networks. In *Proceedings of the 2nd ACM Workshop on Online Social Networks (WOSN'09)*. ACM, New York, NY, 7–12. DOI: http://dx.doi.org/10.1145/1592665.1592668

John Krumm. 2007. Inference attacks on location tracks. In *Proceedings of the 5th International Conference on Pervasive Computing (PERVASIVE'07)*. 127–143. http://dl.acm.org/citation.cfm?id=1758156.1758167

Takeshi Kurashima, Tomoharu Iwata, Go Irie, and Ko Fujimura. 2010. Travel route recommendation using geotags in photo sharing sites. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, 579–588. DOI: http://dx.doi.org/10.1145/1871437.1871513

Min-Joong Lee and Chin-Wan Chung. 2011. A user similarity calculation based on the location for social network services. In *Proceedings of the 16th International Conference on Database Systems for Advanced Applications, Volume Part I (DASFAA'11)*. 38–52. http://dl.acm.org/citation.cfm?id=1997305.1997313

Erich L. Lehmann and Joseph P. Romano. 2005. *Testing Statistical Hypotheses* (3rd ed.). Springer, New York, NY.

Roy Levin and Yaron Kanza. 2014. Stratified-sampling over social networks using MapReduce. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD'14)*. ACM, New York, NY, 863–874. DOI: http://dx.doi.org/10.1145/2588555.2588577

Quannan Li, Yu Zheng, Xing Xie, Yukun Chen, Wenyu Liu, and Wei-Ying Ma. 2008. Mining user similarity based on location history. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS'08)*. ACM, , New York, NY, Article No. 34. DOI: http://dx.doi.org/10.1145/1463434.1463477

Sitong Liu, Guoliang Li, and Jianhua Feng. 2012. Star-join: Spatio-textual similarity join. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM'12)*. ACM, New York, NY, 2194–2198. DOI: http://dx.doi.org/10.1145/2396761.2398600

C. T. Lu, D. Chen, and Y. Kou. 2003. Algorithms for spatial outlier detection. In *Proceedings of the 3rd IEEE International Conference on Data Mining*. IEEE, Los Alamitos, CA, 597–600.

Haiping Ma, Huanhuan Cao, Qiang Yang, Enhong Chen, and Jilei Tian. 2012. A habit mining approach for discovering similar mobile users. In *Proceedings of the 21st International Conference on World Wide Web*. ACM, New York, NY, 231–240. DOI: http://dx.doi.org/10.1145/2187836.2187868

Christian Matyas and Christoph Schlieder. 2009. A spatial user similarity measure for geographic recommender systems. In *Proceedings of the 3rd International Conference on GeoSpatial Semantics (GeoS'09)*. 122–139. DOI: http://dx.doi.org/10.1007/978-3-642-10436-7_8

Grant McKenzie, Benjamin Adams, and Krzysztof Janowicz. 2013. *A Thematic Approach to User Similarity Built on Geosocial Check-ins*. Springer International Publishing, Cham, Switzerland, 39–53.

Marti Motoyama and George Varghese. 2009. I seek you: Searching and matching individuals in social networks. In *Proceedings of the 11th International Workshop on Web Information and Data Management (WIDM'09)*. ACM, New York, NY, 67–75. DOI: http://dx.doi.org/10.1145/1651587.1651604

Arvind Narayanan and Vitaly Shmatikov. 2009. De-anonymizing social networks. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy (SP'09)*. IEEE, Los Alamitos, CA, 173–187. DOI: http://dx.doi.org/10.1109/SP.2009.22

Aviv Nisgav and Boaz Patt-Shamir. 2011. Finding similar users in social networks. *Theory of Computing Systems* 49, 4, 720–737.

Sarana Nutanong, Edwin H. Jacox, and Hanan Samet. 2011. An incremental Hausdorff distance calculation algorithm. *Proceedings of the VLDB Endowment* 4, 8, 506–517. DOI: http://dx.doi.org/10.14778/2002974.2002978

Barak Pat, Yaron Kanza, and Mor Naaman. 2015. Geosocial search: Finding places based on geotagged social-media posts. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, Los Alamitos, CA, 231–234.

Elie Raad, Richard Chbeir, and Albert Dipanda. 2010. User profile matching in social networks. In *Proceedings of the 2010 13th International Conference on Network-Based Information Systems (NBIS'10)*. IEEE, Los Alamitos, CA, 297–304. DOI: http://dx.doi.org/10.1109/NBiS.2010.35

Eliyahu Safra, Yaron Kanza, Yehoshua Sagiv, Catriel Beeri, and Yerach Doytsher. 2010. Location-based algorithms for finding sets of corresponding objects over several geo-spatial data sets. *International Journal of Geographical Information Science* 24, 1, 69–106. DOI: http://dx.doi.org/10.1080/13658810802275560

Hanan Samet. 1984. The quadtree and related hierarchical data structures. *ACM Computing Surveys* 16, 2, 187–260. DOI: http://dx.doi.org/10.1145/356924.356930

Hanan Samet. 2005. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Morgan Kaufmann, San Francisco, CA.

Angela Schwering. 2008. Approaches to semantic similarity measurement for geo-spatial data: A survey. *Transactions in GIS* 12, 1, 5–29. DOI: http://dx.doi.org/10.1111/j.1467-9671.2008.01084.x

Vivek Sehgal, Lise Getoor, and Peter D. Viechnicki. 2006. Entity resolution in geospatial data integration. In *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems*. ACM, New York, NY, 83–90.

Armin Stahl and Thomas Gabel. 2003. Using evolution programs to learn local similarity measures. In *Proceedings of the 5th International Conference on Case-Based Reasoning: Research and Development (ICCBR'03)*. 537–551. http://dl.acm.org/citation.cfm?id=1760422.1760465

Torsten Suel. 2009. Geo-targeted Web search. In *Encyclopedia of Database Systems*. Springer, 1251–1255.

Leong Hou U, Kyriakos Mouratidis, Man Lung Yiu, and Nikos Mamoulis. 2010. Optimal matching between spatial datasets under capacity constraints. *ACM Transactions on Database Systems* 35, 2, Article No. 9. DOI: http://dx.doi.org/10.1145/1735886.1735888

Leong Hou U, Man Lung Yiu, Kyriakos Mouratidis, and Nikos Mamoulis. 2008. Capacity constrained assignment in spatial databases. In *Proceedings of the 2008 ACM SIGMOD International Conference*

on Management of Data (SIGMOD'08). ACM, New York, NY, 15–28. DOI:http://dx.doi.org/10.1145/1376616.1376621

Hao Wang, Manolis Terrovitis, and Nikos Mamoulis. 2013. Location recommendation in location-based social networks using user check-in data. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, New York, NY, 374–383. DOI:http://dx.doi.org/10.1145/2525314.2525357

Kilian Q. Weinberger and Lawrence K. Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research* 10, 207–244. http://dl.acm.org/citation.cfm?id=1577069.1577078

Xiangye Xiao, Yu Zheng, Qiong Luo, and Xing Xie. 2010. Finding similar users using category-based location history. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, New York, NY, 442–445. DOI:http://dx.doi.org/10.1145/1869790.1869857

Mao Ye, Peifeng Yin, and Wang-Chien Lee. 2010. Location recommendation for location-based social networks. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS'10)*. ACM, New York, NY, 458–461. DOI:http://dx.doi.org/10.1145/1869790.1869861

Josh Jia-Ching Ying, Eric Hsueh-Chan Lu, Wang-Chien Lee, Tz-Chiao Weng, and Vincent S. Tseng. 2010. Mining user similarity from semantic trajectories. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks*. ACM, New York, NY, 19–26.

Nicholas Jing Yuan, Fuzheng Zhang, Defu Lian, Kai Zheng, Siyu Yu, and Xing Xie. 2013. We know how you live: Exploring the spectrum of urban lifestyles. In *Proceedings of the 1st ACM Conference on Online Social Networks*. ACM, New York, NY, 3–14.

Yu Zhang, Youzhong Ma, and Xiaofeng Meng. 2014. Efficient spatio-textual similarity join using MapReduce. In *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Volume 01 (WI-IAT'14)*. IEEE, Los Alamitos, CA, 52–59. DOI:http://dx.doi.org/10.1109/WI-IAT.2014.16

Vincent Wenchen Zheng, Yu Zheng, and Qiang Yang. 2009. Joint learning user's activities and profiles from GPS data. In *Proceedings of the 2009 International Workshop on Location Based Social Networks (LBSN'09)*. ACM, New York, NY, 17–20. DOI:http://dx.doi.org/10.1145/1629890.1629894

Yu Zheng and Xing Xie. 2010. Learning location correlation from GPS trajectories. In *Proceedings of the 2010 11th International Conference on Mobile Data Management (MDM'10)*. IEEE, Los Alamitos, CA, 27–32. DOI:http://dx.doi.org/10.1109/MDM.2010.42

Ge Zhong, Ian Goldberg, and Urs Hengartner. 2007. Louis, Lester and Pierre: Three protocols for location privacy. In *Proceedings of the 7th International Conference on Privacy Enhancing Technologies (PET'07)*. 62–76. http://dl.acm.org/citation.cfm?id=1779330.1779335

Yuan Zhong, Nicholas Jing Yuan, Wen Zhong, Fuzheng Zhang, and Xing Xie. 2015. You are where you go: Inferring demographic attributes from location check-ins. In *Proceedings of the 8th ACM International Conference on Web Search and Data Mining*. ACM, New York, NY, 295–304.

Received December 2015; revised August 2016; accepted February 2017