Discovering Personal Gazetteers: An Interactive Clustering Approach

Changqing Zhou, Dan Frankowski, Pamela Ludford, Shashi Shekhar, Loren Terveen
Department of Computer Science and Engineering
University of Minnesota
200 Union ST SE, 4-192
Minneapolis, MN 55414
{czhou, dfrankow, ludford, shekhar, terveen}@cs.umn.edu

ABSTRACT

Personal gazetteers record individuals' most important *places*, such as home, work, grocery store, etc. Using personal gazetteers in location-aware applications offers additional functionality and improves the user experience. However, systems then need some way to acquire them.

This paper explores the use of novel semi-automatic techniques to discover gazetteers from users' travel patterns (time-stamped location data). There has been previous work on this problem, e.g., using ad hoc algorithms [13] or K-Means clustering [4]; however, both approaches have shortcomings. This paper explores a deterministic, density-based clustering algorithm that also uses temporal techniques to reduce the number of uninteresting places that are discovered. We introduce a general framework for evaluating personal gazetteer discovery algorithms and use it to demonstrate the advantages of our algorithm over previous approaches.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Clustering

General Terms

Algorithms, Human Factors

Keywords

Personal gazetteer, Personal places, Location-aware, Spatio-temporal clustering, GPS

1. INTRODUCTION

As mobile devices become *location-aware*, they offer the promise of powerful new applications, such as location-enhanced instant messaging [10] and *digital graffiti* systems [5, 6]. However, fulfilling these promises requires overcoming a number of tough challenges. Our research concentrates on one of these problems, the acquisition and use of personal gazetteers.

A *personal gazetteer* records places that are meaningful for a specific person. While different applications may require different information about places, for us the essential requirements are a textual label and some sort of geometric representation, e.g., a point, a set of points, or a region. Thus, my personal gazetteer might include places such as:

- *Espresso Expose* - latitude: $44^o5702'$, longitude: $93^o1548'$.
- *Home* - lattitude: 445698', longitude: $93^o1527'$.

We have argued elsewhere [11] for the advantages of personal gazetteers based on a notion of place, rather than just physical location. For example, there may be a complex relationship between what a person considers a place and physical location: "Target" might refer to any one of a chain of discount stores within a metropolitan area. Further, the descriptions people use to refer to places – for example, "the coffeeshop", "Espresso Expose", or "the off-brand coffee place next to Big 10" – vary depending on who the descriptions are produced for and the purpose for which they are produced. More generally, research in *environmental psychology* explores how people naturally structure their experience around personally and socially meaningful places - home, office, school, church, coffeeshop, pub, etc. [8] [9] [12]

Once we commit to building systems around users' personal places, we face a new problem: how is a system to acquire personal gazetteers? This paper takes an *interactive discovery* approach to this problem. Specifically, we developed a clustering algorithm that discovers a user's personal gazetteer from the user's spatiotemporal histories, i.e., time-stamped location data. We then embed this algorithm in an interactive system that lets users visualize and confirm, modify, or reject the places discovered by the system.

The remainder of the paper is organized as follows. We first review previous algorithms for discovering personal places and identifies several shortcomings with the algorithms. We then detail our clustering algorithm, including an important temporal pre-processing step. Next, we describe our framework for evaluating personal gazetteer discovery algorithms. We then presents the larger discovery system in which the algorithm is embedded, illustrate how it works on real data (time-stamped GPS readings collected from a GPS-enabled mobile phone), and use the evaluation

GIS'04, November 12–13, 2004, Washington, DC, USA.

framework to compare its performance to the widely used K-Means clustering algorithm. After briefly describing our new research, we close with a brief summary.

2. RELATED WORK

Various approaches could be used to acquire a person's places for use by a sysem. For example, a person could write down a list of places, with or without using a map. However, a list produced from memory might be inaccurate and incomplete. Further, to make the places usable by an application, the position of each place in an earth coordination system would have to be determined somehow and entered.

An alternative is to take an assisted or *interactive discovery* approach. Different researchers have explored different variants of this approach.

2.1 An Exploratory Approach

The comMotion system [13] consists of a device that constantly takes GPS readings. Periodically, the GPS signal is "lost". The loss of the signal is interpreted as a significant cue, namely that a building has been entered. The system maintains a history of readings, and when the signal has been lost within a given radius on three different occasions the agent infers that this location (building) is interesting. When locations are discovered, users are prompted to provide a name; they can do this either immediately or else later while viewing the location on a map. Of course, users also may judge a discovered location to be uninteresting and tell the system to ignore it. Once a location has been discovered and accepted by a user, the user can associate a to-do list with it – a prototypical example is associating a shopping list with a supermarket.

Since the cue for identifying a location is the loss of the GPS signal, only locations such as buildings can be found. Some meaningful places (such as a park or sidewalk cafe) may not cause any GPS signal loss, and thus cannot be discovered. Conversely, in so-called "urban canyons" between tall buildings, GPS signals are often weak and unreliable, which could trigger false discoveries. Further, the use of a fixed radius for delimiting places may be problematic: the size (and shape) of places like a shopping mall, a coffee shop, one's home, and one's place of work can vary widely.

2.2 K-Means Clustering Approaches

Ashbrook and Starner [4] used the well-known K-Means clustering algorithm to learn a user's significant locations from location history data. K-Means is an effecient iterative clustering algorithm. It minimizes an error term which is the sum of squared distances of each point to its cluster center, a mean vector. In formal notation, the error term to be minimized is

$$E = \sum_{i=1}^{c} \sum_{x \in C_i} d(x, m_i),$$

where m_i is the center of cluster C_i, and $d(x, m_i)$ is the Euclidean distance between a point x and m_i.

The algorithm initially assigns all points to a predefined number of clusters randomly. Then it iterates through each point, finds the cluster center nearest that point, and assigns the point to the cluster that the center belongs to. This iteration is repeated until the error term is deemed small or not decreasing much.

K-Means clustering has several drawbacks for detecting a user's places. First, it needs the number of clusters before clustering begins. This could be difficult for users since in general they would not know how many places they frequent. Second, all points are included in the final clustering results, which makes the results quite sensitive to noise. A single noisy or uninteresting location reading far from other points can pull a cluster center toward it much more than it should, because the squared-distance error term heavily weights distant outliers. Third, the K-Means algorithm is non-deterministic: the final clustering depends on the initial random assignment of points to clusters.

2.3 Density-based Clustering Approaches

Density-based clustering uses the density of local neighborhoods of points [7]. There are two parameters used to define density: Eps, the radius of a circle, and $MinPts$, the minimum number of points within that circle. Density-based clustering also uses some notion of the connectivity of that neighborhood, whose points eventually form a cluster. Points that are not in any clusters by the end are deemed noise. Each cluster has a considerably higher density of points than areas outside of the cluster. An example of density-based clustering is shown in Fig. 1.

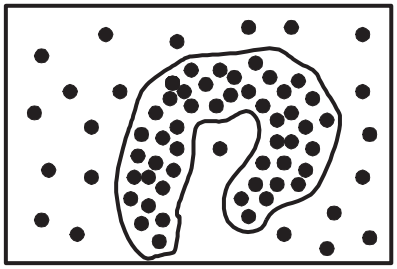


Figure 1: A density-based approach forms a cluster where point density is high.

Many of the limitations of the K-Means clustering approach can be overcome with a density-based clustering approach.

First, it can discover clusters of arbitrary shape. This is a significant improvement over K-Means, which favors symmetric shaped clusters (circles and spheres). There is no reason to believe a person's places are circularly shaped. For example, while a commuter's location history points at a gas station may form a somewhat round shape, a student's location history points on a university campus might easily form an irregular shape.

Second, noise, outliers, or simply unusual points are less likely to participate in the final clustering results. A user may take a trip to the airport only once in two years; he may travel to his boss' house only once in a lifetime (once more than he wanted!); or his GPS device may show a few points far from any actual location he visited. Such points are unusual, and perhaps should be ignored. Fittingly, they may generate few enough points to be discarded because they do not meet the density requirement of a density-based algorithm. In K-Means, they are never ignored; these points will pull cluster centers toward them.

Third, although density-based algorithms require density parameters (e.g., Eps and $MinPts$) as input, these parameters are less likely to need to change within a particular

application. For example, to discover personal gazetteers at the metropolitan level, Eps and $MinPts$ can be predetermined and thus no input is required from the user. The number of clusters in K-Means must be specified per user.

Finally, the density-based algorithms we describe always produce the same clustering given the same input. Anyone who has tried clustering that is sensitive to initial conditions can feel joy at the taming of a disturbing randomness.

In summary, advantages of a density-based algorithm over K-Means are

1. Allows clusters of arbitrary shape.
2. Robustly ignores outliers, noise, and unusual points.
3. Easier to choose reasonable parameter values for a personal gazetteer application that do not depend on the user.
4. Deterministic results.

2.4 DBSCAN and its limitations

DBSCAN is a representative density-based algorithm [7] [14]. One issue with it is that it is very sensitive to the parameters Eps and $MinPts$. For some Eps and $MinPts$, the algorithm will generate a large number of points within its density definition, each of which could be further used to generate its own density-reachable points. In such cases, it will use a lot of memory and slow down considerably.

Our experience with these performance problems led us to develop a different density and join-based clustering algorithm: *DJ-Cluster*. Where DBSCAN uses the connectivity notion of a clique graph, DJ-Cluster instead uses the concept of connected components. This helps DJ-Cluster avoid the performance problems we mentioned.

3. DJ-CLUSTER

The basic idea of DJ-Cluster is as follows. For each point, calculate its *neighborhood*: the neighborhood consists of points within distance Eps, under the condition that there are at least $MinPts$ of them. If no such neighborhood is found, the point is labeled noise; otherwise, the points are created as a new cluster if no neighbor is in an existing cluster, or joined with an existing cluster if any neighbhour is in an existing cluster.

Definition *(density-based neighborhood of a point)* The density-based neighborhood N of a point p, denoted by $N(p)$, is defined by

$$N(p) = \{q \in S \mid dist(p, q) \leq Eps\}$$

where S is the set of all points, q is any point in the sample, Eps is the radius of a circle around p that which defines the density, and $MinPts$ is the minimum number of points required in that circle.

Definition *(density-joinable)* $N(p)$ is density-joinable to $N(q)$, denoted as $J(N(p), N(q))$, with respect to Eps and $MinPts$, if there is a point o such that both $N(p)$ and $N(q)$ contain o. A density-joinable relation is illustrated in Fig. 2.

Definition *(DJ Cluster)* The density and join-based cluster C is defined as follows:

$$\forall p \in S,\ \forall q \in S\ ,\ \exists N(p), N(q)\ such\ that\ \exists J(N(p), N(q))$$

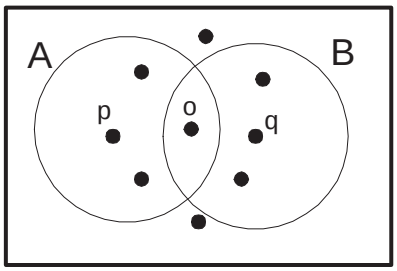


Figure 2: Clusters A and B are *density-joinable* because of the point o.

3.1 Temporal Data Pre-processing

As we tried out the algorithm on real user location data (time-stamped latitude-longitude data obtained from a GPS device), we saw that it found a number of clusters that were not interesting; these included frequent traffic stops on local roads and at pedestrian crossings on campus. These clusters formed because, given a large enough sample of location data, many points will obtained at such frequent stops.

Thus, to improve the performance of the algorithm, we added some temporal pre-processing. Many data filtering techniques could be used to eliminate uninteresting location data. Here are two we used.

First, GPS receivers return not just latitude and longitude, but also a *speed*, estimated by the distance traveled between consecutive readings. We exploited this information to eliminate GPS readings with speeds greater than 0. This removes many GPS readings collected while driving, which did not interest us. It is interesting to note that, in practice, the algorithm still can discover walking or biking paths, which turn out to contain a significant number of readings with speed 0. Since these represent locations where someone frequently stops, such as a traffic light or stop sign or a pleasant park bench, these actually are good candidates to form the significant points of a path.

Second, we eliminated a GPS reading if it was within a small distance of the previous reading. This reduces the amount of data that needs to be processed by the algorithm, therefore speeding it up. Another reason we eliminated these almost-stationary GPS readings is that we had an intuition that indoor places and outdoor places should be represented by similar sets of points. In our trials, we set the mobile phone to take a GPS reading every minute. When you stay at an outdoor location for a long time, the system collects many GPS readings. When you stay at an indoor location for a long time, the system collects few readings, because GPS generally cannot fix indoors. This might put the indoor places at a disadvantage in the clustering process.

It might seem like the application of these two steps would leave no points (!), but there are still plenty of readings left to work with. First, GPS readings are noisy enough that they can have speed 0 and still not be near each other. Second, often the most important places to a person are those that he or she revisits frequently, and each revisit increases the chance that a new, sufficiently-different-to-be-retained reading will be taken.

After describing the temporal pre-processing, we now can state the complete algorithm.

Note that the algorithm has the following properties:

Algorithm 1 DJ-Cluster

1: Perform all temporal data pre-processing.
2: Select an unprocessed point p from sample S.
3: **if** p is null **then**
4: Return
5: **end if**
6: Compute the density-based neighborhood $N(p)$ of a point p wrt Eps and $MinPts$.
7: **if** $N(p)$ is null **then**
8: Label p as noise.
9: **else if** $N(p)$ is density-joinable to at least one existing cluster **then**
10: Merge $N(p)$ and all the density-joinable clusters.
11: **else**
12: Create a new cluster C based on $N(p)$.
13: **end if**
14: Return to step 2.

1. Every point is in one cluster or is ignored as noise.
2. There is always at least one point in each cluster.
3. The algorithm partitions the input into non-hierarchical clusters.
4. The clusters do not overlap.

3.2 Determinism of DJ-Cluster

We now present the proof that DJ-Cluster is deterministic.

THEOREM 3.1. *DJ-Cluster produces one unique clustering.*

PROOF. Suppose $R(p, q)$ is a relation that is true iff p and q are points in the same cluster. We show that R is an equivalence relation. That is, it is reflexive, symmetric, and transitive.

First, by inspection, R is reflexive (a point is in its own cluster) and symmetric (if p is in q's cluster, then q is in p's cluster).

Suppose we are given $R(p, q)$ and $R(q, s)$. These points must have been processed by the algorithm in some order. Suppose the last point processed was p. We know q is in p's neighborhood because $R(p, q)$, and that q and s are in the same cluster because $R(q, s)$. Thus, p will be merged into the same cluster as q and s, so $R(p, s)$.

Suppose instead that the last point was q. Then both p and s will be in q's neighborhood, and the cluster or clusters with p and s will all be merged, and again $R(p, s)$.

Finally, if the last point processed was s, this case is just like p. This proves that R is transitive.

Since R is reflexive, symmetric, and transitive, it is an equivalence relation, which partitions a space uniquely (up to equivalence classes). Thus, the DJ-Cluster clustering is unique. □

3.3 *Eps* and *MinPts*

The combination of Eps and $MinPts$ determines the density of the location neighbours and thus the size and shape of the clusters. To discover smaller, "skinnier" and a larger number of clusters, one can decrease both parameters.

More pragmatically, the values of Eps and $MinPts$ to DJ-Clustering may be determined by specific applications. In an application to discover personal places from GPS data, Eps may be set to approximate the uncertainty in GPS readings, e.g., to 20 meters. Suitable values for $MinPts$ range from 3 to 10; higher values mean that clusters must be more dense to form. In general, this will have the effect of increasing the "precision" of the discovered places while decreasing the "recall" (see discussion below). To consider another example, in an application to discover infectious disease breakout centers within a state, an appropriate value for Eps might be one mile, and 5 might be an appropriate value for $MinPts$.

3.4 Computational Complexity

Due to our small GPS datasets, the current DJ-Clustering is a main-memory implementation. We can analyze it in two steps.

First, temporal pre-processing is $O(n)$. Computing the neighborhood of a point is $O(n^2)$ without a spatial index, or $O(n \log n)$ with an R-tree index.

In the second step, the major cost is the join computation for each point's neighborhood with existing clusters. This is $O(n^2)$ without a spatial index, or $O(n \log n)$ with an R-tree index.

Thus overall, the complexity of the algorithm is $O(n \log n)$ with an R-tree index.

4. EVALUATION FRAMEWORK

Thus far, we have motivated the problem of discovering a personal gazetteer and presented the DJ-Cluster algorithm to address this problem. We further argued that DJ-Cluster is better suited for the personal gazetteer discovery problem than several alternatives. However, as yet, we have offered no way to formalize this argument. In other words, we must answer the question: *how should personal gazetteer discovery algorithms be evaluated?*

We think that standard information retrieval methods and metrics provide an appropriate starting point. Let's review briefly. To evaluate the performance of a retrieval engine, a corpus of documents is first selected. A corpus might consists of a large number of articles from the Wall Street Journal, for example. Then a set of queries is produced: the intention here is to model realistic information needs within a domain. So, for example, a representative query might be: *What is the best way to ensure the safety of the U.S. beef supply?* In the next step, domain experts determine which documents in the corpus are relevant to (or serve as answers for) each query. These documents serve as the baseline or "gold standard" for evaluating the results returned by any given search engine.

Two major metrics are traditionally used, precision and recall. *Precision* measures the proportion of results returned by a search engine for a query that were in the "gold standard". *Recall* measures the proportion of documents in the "gold standard" for a query that were returned by a search engine.

While we find this framework largely suitable, we must revise it in several ways. First, since we are discovering *personal* places for individuals, each user of the system must define his or her own baseline, i.e., the set of places they find meaningful. Second, since there is no a priori corpus of places, we must take into account that people may not be able to think of all the places they care about when producing their baseline. Thus, our metrics must include a way to measure *discovery* of interesting places that were not actually in the baseline set. Third, the input to the discovery algorithm consists of time-stamped location data for

a person, e.g., latitude-longitude coordinates obtained from a GPS device. To increase the likelihood that places that matter to a person are represented in this dataset, it should be collected over a period of time, e.g., several weeks.

We now are in a position to formalize our methods and metrics for evaluating personal gazetteer discovery systems.

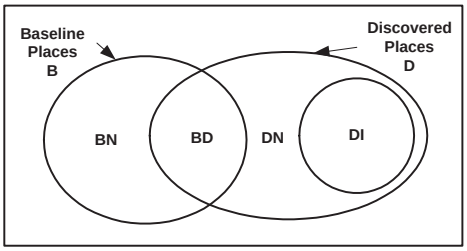


Figure 3: Sets of Baseline and Algorithm-Discovered Places

To evaluate a personal gazetteer discovery algorithm:

1. Collect a personal location dataset.
2. Collect a person's baseline places (call this set B). We obtained baseline places through a combination of the following two approaches:
   (a) *Unassisted*: The person logs the places he or she visits during the data collection period.
   (b) *Assisted*: At the end of the data collection period, the person's raw location data is presented on a map. The person uses the map as an aid to remember more places.
3. Run the algorithm on the personal location dataset to discover places in the person's gazetteer (call this set of places D).
4. Present the discovered places to the user on a map. Ask the person to match their baseline places to the discovered places, obtaining the following sets (figure 3 summarizes the various sets of places):
   - *BD*: Baseline places that were discovered.
   - *BN*: Baseline places that were not discovered.
   - *DI*: Discovered places that are interesting and significant to the person.
   - *DN*: Discovered places that are not interesting to the person.
5. Compute the *Precision*, *Recall*, and *SurpriseFactor* metrics, defined as follows:

$$recall = \frac{|BD|}{|B|}$$

$$precision = \frac{|BD|}{|D|}$$

$$SurpriseFactor = \frac{|DI|}{|D|}$$

Note that it makes sense to sum the *precision* and $SurpriseFactor$ scores for an algorithm. This derived quantity represents how many of the discovered places were "good", that is, either were in the user's baseline or were judged interesting and significant.

5. APPLYING AND ILLUSTRATING OUR APPROACH

To actually try out and evaluate the algorithms we have described, we must embed them in a working system and collect real data. We have constructed a suitable system, which we describe here. We have only completed the collection and analysis of data for a few people, including the first author of this paper, Zhou. Thus, we use Zhou's data for illustrative purposes here. We are in the process of analyzing data from an experiment where we collected data from 25 people; we describe this briefly in the Future Work section.

In this section, we describe our experiences with data collection and baseline place discovery for Zhou. We ran K-Means and DJ-Cluster on Zhou's location history data and discovered his places. The results were evaluated using the recall, precision and surprise factor metrics.

5.1 Personal Gazetteer Discovery System

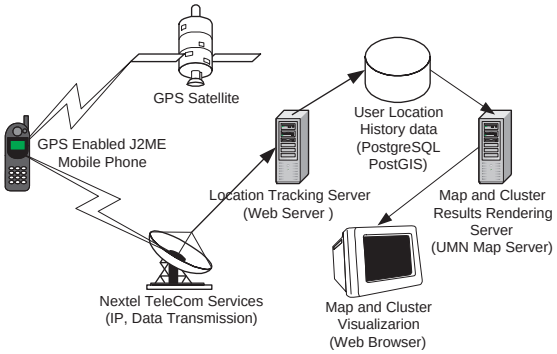


Figure 4: Components of a Personal Gazetteer Discovery System

The personal gazetteer discovery system consists of three major components: location sensing, location data storage, and visualization. These are illustrated in figure 4. To sense location data, we used a GPS-enabled phone (the Motorola i88s) and Nextel's phone service. To store location data, we set the phone to use a free service, AccuTracking [1]. To visualize the maps, we used the University of Minnesota MapServer [3], which supports Open GIS Consortium (OGC) [2] standard spatial queries and operations. For computation, we stored GPS data in a postgreSQL database with PostGIS extensions.

5.2 Collecting a Personal Location Dataset

The first author of this paper carried a GPS-enabled mobile phone for three weeks as he went about his daily activities in the Minneapolis – Saint Paul metropolitan area in the United States. His normal transportation mode was driving a personal car. His routine included commuting to work, frequent visits to the University of Minnesota campus, and various errands.

He attempted to keep the phone with him and on at all times. The phone ran the Accutracking service, which was configured to take a GPS reading every minute. During the 20-day experiment period, 3,469 GPS readings were collected. On average, this was about 173 locations per day, or nearly three hours worth of location data. These location data are visualized on the map in figure 5.

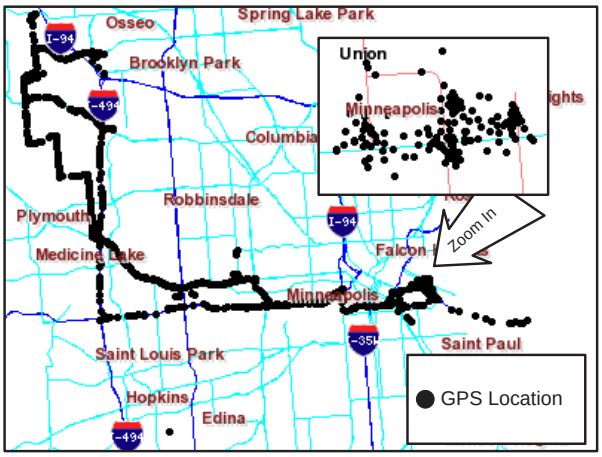


Figure 5: Zhou's Personal Location Dataset

Visualization Only	EECS, Coffee Shop, Parking I, Parking II
Recording Only	Cub, Tea House, Dentist, Pediatrician, BestBuy, Tax Preparer, Bank, Grocery Store, and University
Both	Home, Work, Daycare, Community Center, United Noodle

Table 1: Zhou's Baseline Places

5.3 Collecting Baseline Places

The baseline places for Zhou are shown in table 1 and (as black squares) in figure 6. Note that he logged most of the places while his personal location dataset was being collected, but added four more paces after looking at the raw location data presented in figure 5.

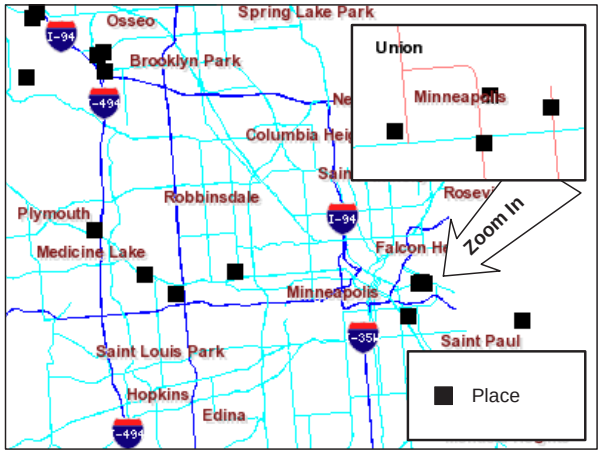


Figure 6: Zhou's Baseline Places

5.4 Results

We ran both a standard K-Means algorithm (that we implemented) and DJ-Cluster on Zhou's personal location dataset. For DJ-Cluster, we set $MinPts = 10$ and $Eps = 10$. Figure 7 shows the baseline places and the results obtained by K-Means; figure 8 shows the baseline places and the results obtained by DJ-Cluster.

	K-Means	DJ-Cluster
Baseline(B)	18	18
Discovered Places(D)	21	21
Baseline-coverage(BD)	5	15
Not in baseline but interesting(DI)	0	3
Not in baseline and not interesting(DN)	16	3
Recall	5/18 or 28%	15/18 or 83%
Precision	5/21 or 24%	15/21 or 71%
SurpriseFactor	0/21 or 0%	3/21 or 14%

Table 2: Evaluation of K-Means and DJ-Cluster

For each algorithm, Zhou matched the baseline places to the discovered places and then judged each non-baseline place that was discovered as either interesting or not. This enabled us to compute the evaluation metrics. Table 2 presents the results. Of course, these results – for just one person's data, with that person being an author of this paper – do not prove that DJ-Cluster is superior to K-Means. However, our previous analysis of the shortcomings of K-Means makes us confident that DJ-Cluster's superiority will hold up in a more systematic evaluation. We next discuss in more detail how each algorithm performed.

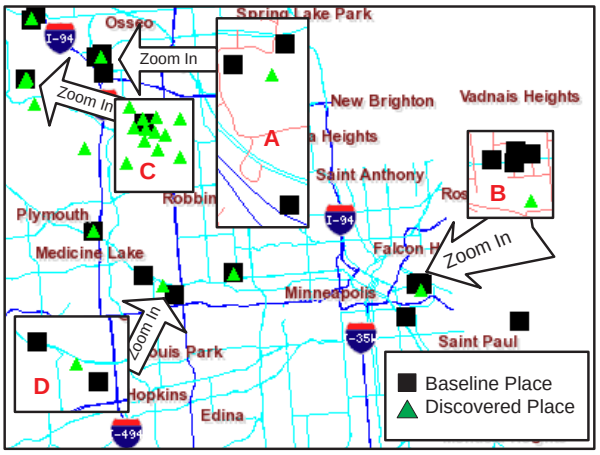
5.5 K-Means

Figure 7 shows that at a city scale, the places discovered by K-Means are geographically close to the baseline places. However, the zoom-ins in the figure reveal that many discovered places are at least a couple of city blocks away from the baseline places, and the mapping between baseline and discovered places is quite inexact. For example, zoom-in B contains four distinct baseline places in the campus area. Yet the K-Means algorithm created only one cluster in that area, and that cluster is 4-5 blocks away from the nearby baseline places. Similarly, no clusters were created for other baseline places such as Dentist, Cub Foods, and United Noodle because there were relatively few location data for these places, and they happened to be far away from the initial randomly selected cluster centers.

On the other hand, zoom-in C shows that K-Means created about 10 clusters for just one baseline place, Home. This is because a large proportion (more than half) of all the location data were near Home, so randomly selecting points for the initial cluster centers led to many of these points being selected.

5.6 DJ-Cluster

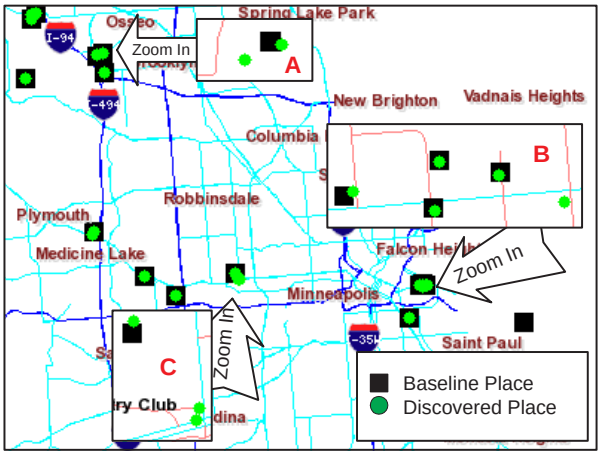
Compared to K-Means, DJ-Cluster shows discovered places closer to each baseline place; in fact, they often actually overlap. See zoom-in B of figure 8 for example. DJ-Cluster also discovered some interesting places that were not in the person's baseline places, such as Chipotle, WalMart, and an ice arena. This offers some evidence for our claim that an assisted approach to place acquisition offers advantages over a purely user-driven one. Finally, DJ-Cluster minimized the number of spurious places discovered around



A: A cluster is formed from Cub Foods, community center, and BestBuy
B: A cluster is formed from campus places, United Noodle, and tax agency.
C: Many clusters are generated from home
D: A cluster is formed from the dentist's and Tea House

Figure 7: K-Means Results and Baseline Places

traffic lights and pedestrian stops.



A: Community center: swimming pool and ice arena
B: Campus: parking I, parking II, EE/CS, coffee shop, and Chipotlet
C: Work: work and 2 traffic stops

Figure 8: DJ-Cluster Results and Baseline Places

6. FUTURE WORK

In our first followup to this study, we seek to better understand how people conceptualize place and how their concepts relate to physical locations, and to evaluate more systematically the accuracy of our DJ-Cluster algorithm. We enlisted 25 subjects to carry GPS-enabled mobile phones for a 3 week period. As in the exploratory study reported here, location readings were taken every minute, and subjects logged their places at the end of each day. After the data collection phase was completed, we applied DJ-Cluster to each subject's personal location dataset to produce a map for each subject. We then conducted a semi-structured interview organized around each subject's logged places and the generated map. We evaluated the accuracy of the discovered places and probed the physical structure of places. We currently are in the process of analyzing the results, which will provide a useful accuracy baseline for personal gazetteer discovery algorithms.

We also are planning to do more complicated analysis of location history datasets. For example, we intend to extend our algorithm to discover travel routes, not just static places. We also will explore *social matching* [15] algorithms, which can bring together two different people based on overlap in their spatiotemporal routines. This could be useful for finding commuting partners, among other applications.

7. SUMMARY

We presented a deterministic clustering algorithm, DJ-Cluster, intended to be used for discovering a user's significant places from location data. Compare to previous algorithms, DJ-Cluster has several important technical advantages: it allows clusters of arbitrary shape; robustly ignores outliers, noise, and unusual points; has more easily chosen parameters; and has deterministic results. We also described several temporal filtering techniques that reduce the number of uninteresting places discovered by the algorithm.

Finally, we presented a framework for evaluating algorithms that tackle the personal gazetteer discovery problem. Using this framework, we presented results that suggest that DJ-Cluster significantly outperforms standard algorithms. Current work is firming up our understanding of the algorithm's accuracy, and planned work will extend it to handle additional challenging and interesting problems.

8. ACKNOWLEDGMENTS

This work was partially supported by grants from the NSF (IIS 03-07459 and IIS 03-08018). It also was partially supported by the Army High Performance Computing Research Center (AHPCRC) under the auspices of the Department of the Army, Army Research Laboratory (ARL) under contract number DAAD19-01-2-0014. The content of this work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred.

9. REFERENCES

[1] AccuTracking Web Site. http://www.accutracking.com.

[2] Open GIS Consortium, Inc. (OGC). http://www.opengis.org.

[3] University of Minnesota MapServer. http://www.mapserver.umn.edu.

[4] D. Ashbrook and T. Starner. Learning Significant Locations and Predicting User Movement with GPS. In *Proceedings of IEEE Sixth International Symposium on Wearable Computing*, 2002.

[5] J. Burrell and G. Gay. E-graffiti: evaluating real-world use of a context-aware system. *Interacting with Computers*, 2001.

[6] F. Espinoza, P. Persson, A. Sandin, H. Nystrm, E. Cacciatore, and M. Bylund. Geonotes: Social and navigational aspects of location-based information systems. In *Proceedings of Ubicomp*, 2001.

[7] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, 1996.

[8] Franck, Karen A., and L. H. Schneekloth, editors. *Ordering space: types in architecture and design*. Van Nostrand Reinhold, 1994.

[9] R. Genereux, L. Ward, and J. Russell. The behavioral component in the meaning of places. *Journal of Environmental Psychology*, 3:43–55, 1983.

[10] W. Griswold, G. Shanahan, S. Brown, R. Boyer, M. Ratto, R. Shapiro, and T. Truong. Activecampus - experiments in community-oriented ubiquitous computing. Technical report, UC San Diego, 2003.

[11] Q. Jones, S. Grandhi, S. Whittaker, K. Chivakula, and L. Terveen. Putting systems into place: A qualitative study of design requirements for location aware community systems. In *Proceedings of CSCW*, 2004.

[12] B. Kramer. Classification of generic places: Explorations with implications for evaluation. *Journal of Environmental Psychology*, 15:3–22, 1995.

[13] N. Marmasse and C. Schmandt. Location-aware information delivery with commotion. In *HUC*, pages 157–171, 2000.

[14] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2:169–194, 1998.

[15] L. Terveen and D. McDonald. Social matching: A framework and research agenda. *ACM Transactions on Computer-Human Interaction*, to appear.