Tracommender – Exploiting Continuous Background Tracking Information on Smartphones for Location-Based Recommendations

Yang Wang, Abdulbaki Uzun, Ulrich Bareth, and Axel Küpper

Telekom Innovation Laboratories, TU Berlin, Service-centric Networking
wangyang.tub@gmail.com, abdulbaki.uzun@telekom.de,
ulrich.bareth|axel.kuepper@tu-berlin.de
http://www.snet.tu-berlin.de/

Abstract. In this paper, we propose *Tracommender*, a context-aware recommender system, which uses background tracking information from smartphones to generate location-based recommendations. Based on the automatically collected data that consist of locations with timestamps, the dwell time at certain locations can be derived in order to use it as an implicit rating for a location-based collaborative filtering. We further introduce two alternative path matching algorithms that utilize continuous location sequences (paths) to compute path patterns between similar users. In addition, in order to overcome the cold-start problem of recommender systems, clustering algorithms are used to calculate so-called *Activity Zones* - locations taken from an existing database of categorized points of interest. Synthesized movement data has been applied to perform evaluations on performance, scalability and precision of an implemented prototype of the proposed recommendation algorithms.

Keywords: location-based services, background tracking, recommendations, path matching.

1 Introduction

With the increasing number of location-based services, context-aware recommender systems become more and more relevant when recommending content items, such as products, restaurants or shops. Contextual data (e.g., location, time of day or weather) is a promising information source to exploit in order to generate more precise recommendations that do not only fit to a user's profile and ratings given to those content items by a community, but also on the contextual situation the user is in.

However, not all kinds of content items are suitable for context-based recommendations. Moreover, the automatic detection of some context parameters (e.g., the mood of a user or companions) turns out to be very difficult (or sometimes impossible) and can only be integrated in the form of manual input, such as a

C. Borcea et al. (Eds.): MobilWare 2012, LNICST 65, pp. 250–263, 2013.

scrollbar where users can adjust the "mood" of a song [1]. Due to the possibility of users providing false information, these types of manual input are no reliable information sources and the scenarios in which those context data is used, seem not very applicable in real business services. Another aspect is that context is often treated as a single and static piece of information, it is not considered as a continuous sequence. However, the former and latter pieces of information in a context sequence may also be useful in order to determine a user's intention.

The location information, on the other hand, is the most important context that fulfills the requirements mentioned above and is therefore very suitable when creating context-aware location-based recommendations. It can be determined in an accurate manner utilizing smartphones and positioning methods like Cell ID, WiFi and GPS [2] making it trustworthy and automatically detectable. Using background tracking data generated by mobile devices as a reliable, relevant and constant information source, a history of user paths (location sequences) can be calculated and used in the recommendation process, which might give a hint on which locations a user might be interested when taking a certain path. Each location on a path can also be enriched by context information that is directly derived by the location information like the location dwell time or weather, in order to provide much more precise recommendations.

In this paper, we propose *Tracommender*, a novel context-aware location-based recommender system that utilizes background tracking information collected by mobile devices via a crowd-sourcing approach in order to provide location recommendations. The system incorporates a hybrid approach including a location-based collaborative filtering algorithm, two alternative path matching methods and an innovative concept of *Activity Zones* to overcome the cold-start problem [3].

The remainder of the paper is organized as follows: First, an overview about related work in the area of context-aware and location-based recommender systems is presented. Section 3 describes the concept of *Tracommender*, including the location-based collaborative filtering method based on location dwell time frequencies, the two alternative path matching approaches, the innovative concept of *Activity Zones* for tackling the cold-start problem and the system architecture. A performance evaluation is done in Section 4, whereas the last section concludes the paper.

2 Related Work

In a world of information overload, recommender systems filter relevant information and provide personalized content item recommendations to users based on their personal background, preferences and interests. Numerous recommendation methods were designed over the years to enhance the preciseness of recommendations. Besides the content-based algorithm, collaborative filtering is one of the most well-known and established recommendation methods [4].

Collaborative filtering uses the previously rated items of a user community as a basis in order to predict content items to the active user. The user-based collaborative filtering approach utilizes the ratings of the active user and the ratings

of other users in order to compute similarities between them. The items of the similar users are then recommended to the active user. In order to increase the performance and quality of user-based collaborative recommendations, Sarwar et al. [5] introduced an item-based collaborative filtering approach. The main idea of this method is that instead of detecting similar users, the similarity of items is calculated based on the ratings given by different users. Two items are considered more similar the more users have rated both of them. After identifying the most similar items, the weighted average of the active user's ratings on these items is used to calculate predictions. Similarities between users or items are measured by using two alternative equations, the *Pearson Correlation Coefficient* and the *Cosine Similarity* measure, which are adopted in Section 3.1.

Traditional recommendation approaches solely focus on recommending items to users without considering the context the user is in. However, thinking of mobile applications and especially location-based services, the contextual situation of a user is an essential factor for providing relevant recommendations. Contextual information can support recommender systems in three possible phases: During the preparation phase, the contextual situation serves as conditions for information filtering, such as in the works of Baltrunas et al., who propose a context-aware item splitting approach for collaborative filtering [6] and the "best context" for music recommendations [1]. In the second phase, context is regarded as a special item processed and filtered by recommendation approaches, such as in the paper of Domingues et al., who use contextual information as virtual items on recommender systems [7]. Finally, during the phase of presenting results, context works as a post-filter to correct inappropriate recommendation results, as proposed by De Carolis et al. [8].

The location information as the most important context is utilized in many previous works like *CityVoyager* [9], *TouristGuide* [10], *Shopper's Eye* [11] or `foursquare.com` in order to provide location-based recommendations. But location is only regarded as a static piece of information independent from other context. None of these approaches recognize the spatial and chronological continuity of the whole context or uses the whole sequence of historical location and other contextual information to derive habits and dependencies. Here, background tracking could be of immense use by continuously recording location information and other context data over time to improve the overall quality of recommendations.

3 Concept of the Tracommender

In this section, we propose our concept called *Tracommender*, which is a portmanteau out of the two words *tracker* and *recommender*. The first word *tracker* describes that background information, such as location and context data is continuously being collected and used for *recommendations*, which is the second word with a rather obvious meaning.

Tracommender uses a hybrid recommendation process including a location-based collaborative filtering algorithm that determines the similarity of users in terms of the locations they have visited. In addition, the system provides two alternative path matching approaches, the *adjacency matrix* and *minimum distance matching* algorithms, which identify path patterns out of historical paths of the nearest neighbours computed by the location-based collaborative filtering method and the current path of the user in order to predict future locations on his current path. The third component of the hybrid approach comprises the concept of *Activity Zones* that define areas clustered by geographic regions offering similar places in high density. The *Activity Zones* tackle the cold-start problem of *Tracommender* when lacking a critical number of path information in the initial phase in order to be able to generate precise recommendations. Last but not least, the system architecture used for the implementation is presented.

3.1 Location-Based Collaborative Filtering

Tracommender uses path similarities calculated out of historical paths in order to predict locations that might be of interest for a user on his current path. Those historical paths can either be generated by the user himself or by other people. Taking only the paths of the active user as a basis for recommendations might produce results that fit to the user's personal movement patterns. However, the amount of candidate paths considered in the recommendation process will be limited. Extending the data basis for recommendation calculation by all paths of all users available will provide more paths, but will weaken the correlation between the user's current path and the historical paths of others.

In order to combine both approaches and determine only paths of those users relevant to the active user, collaborative filtering is utilized as a preperation for the path matching algorithms. Generally, in collaborative filtering, users who rate items similarly are considered as being nearest neighbours. We adopted this paradigm, so that we can collect all users in a set of neareast neighbours that share the same location preferences as the active user. The paths of those neighbours are then used when generating path patterns via the path matching algorithms.

Collaborative filtering usually works with numerical rating values when calculating recommendations. A high numerical rating value represents a user's strong interest towards a certain content item. Locations can also be seen as content items that can be rated. Therefore, we exploited the users' dwell time on specific locations (being a context parameter that is directly derived by the location information) as implicit and automatic feedback to indicate their personal interests and preferences for certain locations. If we consider a user's dwell time on a single location as an implicit rating given by this user to this location, we can construct a *user-location-dwell-time-frequency matrix* similar to a *user-item-rating matrix*, which is used within the collaborative filtering algorithm.

Based on the definition of *term frequency (TF)* [4], which is defined as the result of dividing the occurrence count of a term in a document by the total number of terms in the document, we can define the dwell time frequency $f_{u,l}$ of

user u on location l as the result of dividing the sum of dwell times on location l by the sum of dwell times on all locations belonging to the location set $L(l \in L)$, i.e.,

$$f_{u,l} = \frac{S_l}{\sum_{k \in L} S_k} = \frac{\sum_{t=T_1}^{T_2} n_{t,l}}{\sum_{k \in L} \sum_{t=T_1}^{T_2} n_{t,k}} \tag{1}$$

where S_k is the sum of dwell times on location k. T_1 and T_2 denote the starting and ending time of a given time period; $t = \{T_1, \ldots, T_2\}$, on the other hand, denotes a specified point of time during the given time period. The interval and unit of the time period can range from seconds to minutes, which depends on the accuracy of the tracking unit of the system. $n_{t,l}$ is a binary value: $n_{t,l} = 1$ if and only if user u was at location l at moment t, otherwise $n_{t,l} = 0$. Equation (1) indicates that the longer a user has accumulatively stayed at a location than other locations during a time period, the higher rating the user gives to the location. Assume a set of users $U = \{1, \ldots, m\}$ and a location set $L = \{1, \ldots, n\}$ existing in the database, the *user-location-dwell-time-frequency matrix* M can be expressed as:

$$M = \left\{ f_{u,l} | u = \{1, \ldots, m\} \text{ and } l = \{1, \ldots, n\} \right\} \tag{2}$$

The similarity of two users a and b can be calculated with the *Pearson Correlation Coefficient Similarity* or *Cosine Similarity* measure [4]. In the case of dwell time frequency, the two similarity equations are adapted and expressed as

$$sim(a,b) = \frac{\sum_{l \in L} (f_{a,l} - \overline{f_a})(f_{b,l} - \overline{f_b})}{\sqrt{\sum_{l \in L} (f_{a,l} - \overline{f_a})^2} \sqrt{\sum_{l \in L} (f_{b,l} - \overline{f_b})^2}} \tag{3}$$

$$sim(a,b) = cos(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\|\boldsymbol{a}\| \times \|\boldsymbol{b}\|} = \frac{\sum_{l \in L} f_{a,l} f_{b,l}}{\sqrt{\sum_{l \in L} f_{a,l}^2} \sqrt{\sum_{l \in L} f_{b,l}^2}} \tag{4}$$

where $\overline{f_a}$ and $\overline{f_b}$ denote the average dwell time frequency of users a and b on all locations. Having the similarities computed between each user, the historical paths of the nearest neighbours can be used in the path matching process.

In comparison to explicit ratings given by users, the dwell time frequency has several advantages in terms of credibility and density. First, the dwell time frequency is automatically derived from background tracking information making it a very reliable information source. Secondly, users are not required to rate locations manually. This also ensures reliability due to the fact that it is not guaranteed that users will provide true rating values. Furthermore, it supports the user experience, because users are not asked to give ratings all the time at each location on their path. In addition, it tackles the sparsity problem [4] of collaborative filtering, since the automatically calculated dwell time frequency guarantees a high number of ratings (in comparison to the number of manually given ratings), which is essential for a recommendation algorithm to work accurately. Finally, the implicitly given feedback reflects factual interests of the users

rather than a subjective opinion in the form of explicit feedback (user a and b may like a location equally, but rate it differently). Depending on the time spent on a location, the frequency of favourite places will have a higher rating than places temporarily visited by them. However, one drawback of this approach is that it is not distinguished between places that are really favoured by users and places where they are "forced" to spend much time like workplaces. This problem will be addressed in the near future by integrating semantic information (e.g., ontologies about location classification) and a mixture of explicit/implicit feedback into the recommendation process.

The location-based collaborative filtering algorithm enables *Tracommender* to provide location recommendations to a user based on the opinions of like-minded users in the community. Theoretically, this recommendation method can work as a stand-alone service in the system. Having a database with users, locations and the ratings given to those locations computed by the dwell time frequency, the system can provide location recommendations to a user independent from the current path he is on. However, since *Tracommender* does not only care about single locations, but also about sequences of locations, the results provided by the collaborative filtering algorithm are integrated into the path matching process, which is described in Subsection 3.2.

3.2 Path Matching Algorithms

The nearest neighbours computed using the location-based collaborative filtering algorithm build the basis for the path matching approaches described in this section. The *adjacency matrix matching* and *minimum distance matching* algorithms are utilized to find path patterns between the paths of the nearest neighbours and the current path of the active user. These path patterns help to predict the movement of the user in order to recommend him locations that he might like to visit on his path.

Depending on the data model, two different methods can be used. If the location sequences are modeled in a list fashion, their similarity can be expressed as the distance between those two, which is calculated with our minumum distance implementation. Another way to compute the similarity of location sequences is by expressing them as paths in adjacency matrices as explained below.

Adjacency Matrix Matching. Given a finite directed graph, an adjacency matrix is a boolean square matrix that represents the directed edges between vertices of the graph. The edges $E_{i,j}$ of a path P can be expressed as 1 when there is an existing (directed) connection between the vertix i and vertix j and 0 otherwise.

$$E_{i,j} = \begin{cases} 1, (V_i, V_j) \\ 0, (V_i, V_j) \end{cases} \tag{5}$$

The directed graphs of the two paths $P = \{p_1, \ldots, p_m\}$ and $Q = \{q_1, \ldots, q_n\}$ are modeled as G_p and G_q. The adjacency matrix of graph G_p can be expressed as:

$$A = \{a_{i,j} | i, j \in P;\ a_{i,j} = E_{i,j}\} \tag{6}$$

where $a_{i,j} = 1$ when a directed edge (i, j) exists in graph G_p, which is the case when location i can be reached from location j within path P; and $a_{i,j} = 0$ if the directed edge (i, j) does not exist. The adjacency matrix of graph G_q is defined the same way. If the two paths contain different locations, they have to be modeled and matched in a collective set of locations $S = P \cup Q$. Equation 6 can be rewritten as follows:

$$A = \{a_{i,j} | i, j \in S;\ a_{i,j} = E_{i,j}\} \tag{7}$$

where A is a square matrix of dimension k-by-k and $k = |S|$ is the cardinality of set S. The exclusive-or matrix D out of adjacency matrix A of path P and adjacency matrix B of path Q is generated with the logical operation exclusive disjunction on each pair of counterpart entries of the two adjacency matrices, i.e.,

$$D = A \oplus B = \{d_{i,j} | d_{i,j} = a_{i,j} \oplus b_{i,j}\} \tag{8}$$

which represents how many exclusive-or relations the two matrices have in common. Thus, the similarity between the two paths through adjacency matrix matching is defined as:

$$sim(P, Q) = sim(A, B) = 1 - \frac{\sum_{d \in D} d}{\sum_{a \in A} a + \sum_{b \in B} b} \tag{9}$$

The two paths share all their path segments with each other if the similarity equals one, thus they are said to be structurally the same. The two paths are independent if the similarity equals zero.

Example: Two paths $P: \ a \to b \to c \to b$ and $Q: \ b \to a \to c \to b$ are shown in Figure 1a, where a, b, c describe locations or vertices on the path with the grey node as starting point. The adjacency matrices of P and Q and their exclusive-or matrix are expressed as:

$$\mathbf{A} = \begin{pmatrix} 0\,1\,0 \\ 0\,0\,1 \\ 0\,1\,0 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} 0\,0\,1 \\ 1\,0\,0 \\ 0\,1\,0 \end{pmatrix} \qquad \mathbf{D} = \mathbf{A} \oplus \mathbf{B} = \begin{pmatrix} 0\,1\,1 \\ 1\,0\,1 \\ 0\,0\,0 \end{pmatrix} \tag{10}$$

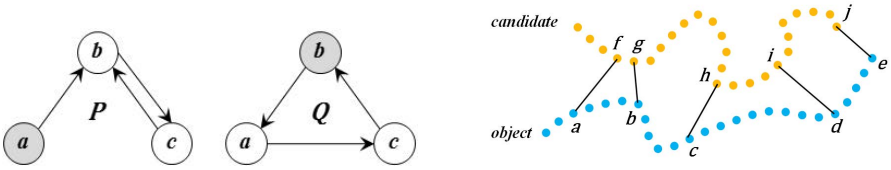
Acoording to equation 9, the similarity of the adjacency matrices is

$$sim(A, B) = 1 - 4/6 = 1/3 \tag{11}$$

Since an adjacency matrix can represent the relations of vertices, it also holds for a mathematical expression of directed graphs. By comparing two adjacency matrices, their similarity can be calculated, which expresses how many numbers of locations two paths have in common in relation to the total number of their locations. Although sometimes, even if two paths do not have any location in common, their locations might be geographically close to each other. An example are the two paths in Figure 1b, which would have a similarity result of 0

when computed with adjacency matrix matching, because they share no common location. But the two paths seem to be quite similar. Therefore, another path matching method capable of measuring the geographical distance between paths of different locations is presented in the following.

Minimum Distance Matching. To also express the similarity of nearby paths that have no locations in common, minimum distance matching selects the nearest paths from a set of known candidates for an object path by calculating the sum of the minimum distances. The distance of a point to a candidate path is defined as the minimal euclidean distance from the point to every point on the path. In Figure 1b, for example, the points a, b, c, d, e on the object path have the best similarity to f, g, h, i, j on the candidate path, when the sum of their minimal euclidean distances is smaller than the sum of other candidate paths. If the sum of these distances from each point on the object path to the candidate path is shorter than to other paths, then the candidate path is regarded as the nearest path to the object path.



(a) Adjacency Matrix (b) Minimum Distance

Fig. 1. Path Matching Algorithm Examples

Two locations $p = (x_1, y_1)$ and $q = (x_2, y_2)$, with x_1 and x_2 as longitude, y_1 and y_2 as latitude, when neglecting height and the spherical equation of the earth's surface for simplicity and scalability reasons, the distance between the two locations can be described by the Euclidean distance:

$$d(p, q) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{12}$$

Given an object path $P = \{p_1, \ldots, p_m\}$ and a candidate path $Q = \{q_1, \ldots, q_n\}$, for each $p_i \in P$ there is a set $\{d(p_i, q) |\ q \in Q\}$ resembling the collection of distances from p_i to every point on the path Q. The sum of the minimum distances from P to Q is defined as

$$\begin{aligned} MD(P, Q) &= \sum_{i=1}^{m} min\{d(p_i, q) |\ q \in Q\} \\ &= \sum_{i=1}^{m} min\{d(p_i, q_1), \ldots, d(p_i, q_n)\} \end{aligned} \tag{13}$$

where $p_i \in P$ and min is a function returning the minimum value in a set.

The sum of the minimum distances is inverse to their similarity meaning that the smaller the sum of minimum distances of two paths are, the more similar they are. The algorithm can be repeated for several candidate paths to find the nearest or most similar paths.

While *adjacency matrix* is stronger related to the structural similarity of two location paths or sequences, the *minimum distance* expresses geographical similarity or proximity of two graphs.

3.3 Activity Zones

Recommender systems inherently suffer from the cold-start problem [3], which basically means that no recommendations can be calculated as long as no relevant data exists yet. Therefore, *Activity Zones* can be created to overcome the cold-start problem for location-based recommendations by clustering existing locations for certain categories of interest.

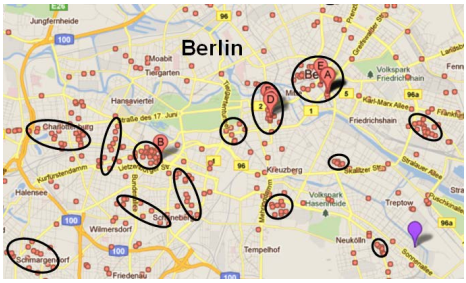


Fig. 2. Shopping Activity Zones in Berlin

Activity Zones are created by applying clustering methods to databases of locations with the same categories like restaurants, shops or theaters. For this purpose, several clustering methods have been analyzed. Nonhierarchical methods are not effective due to the fact that the number of clusters is not known beforehand. Furthermore, a maximum distance and cluster density has to be specified to not generate clusters, which are too big or incoherent. Therefore, the complete linkage or average linkage method has been chosen to create the desired *Activity Zones* of *Tracommender*.

In this way, users' locations are classified to certain categories and recommendations for other locations of the same category can be made. Figure 2 shows the resulting clusters of shopping locations in Berlin on *Google Maps*. Note that not circles but ellipses are used to more accurately describe the clusters.

3.4 System Architecture

Tracommender's system architecture contains several primary components as described in the previous sections working in an offline and an online phase. The

three different blocks (see Figure 3) reflect the major steps in the recommendation process. During the *Crowd-Sourcing Phase*, mainly location data is collected continuously in a background process on the users' mobile terminals. Based on a certain time interval, it is aggregated and sent to the *Tracking Sequence Database* where additional context like dwell time at certain locations are extracted during the *Offline Phase*. The *Location-based Collaborative Filtering* algorithm detects nearest neighbours based on the dwell time frequency of locations, which is then used in the *Path Matching* process. In addition, clustering is being performed in order to create *Activity Zones*. In the *Online Phase*, locations of an *Activity Zone* are recommended if the system detects that the user is in or near to such a cluster. Otherwise, locations are recommended based on the *Path Matching* algorithms.

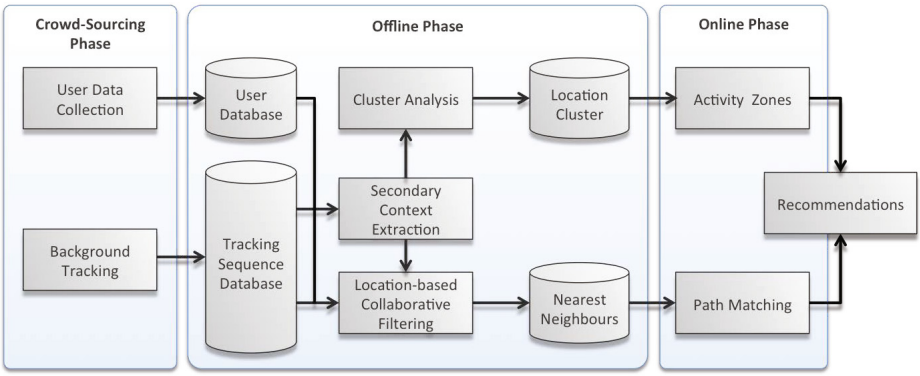


Fig. 3. Tracommender - System Architecture

4 Evaluation

The recommendation approach proposed in this paper is evaluated against precision and performance of the recommendation results computed by the path matching algorithms based on a critical mass of user, location and path information obtained by mobile devices and background tracking information. Since we were currently not able to run a real field test with a big number of mobile devices in a crowd-sourcing approach, we created a simulated crowd-sourcing database including user, location and path information for evaluation and demonstration of our proof-of-concept.

4.1 Simulated Crowd-Sourcing Data

The simulated crowd-sourcing database that is created in order to evaluate the recommendation approaches and algorithms proposed in this paper, consists of 50 users, 240 factual locations obtained from *Google Maps*, and 3837 pieces of background tracking information generated with our *Crowd-sourced Path Simulation Algorithm*. This algorithm is designed to create personalized path records.

For this purpose, a number of locations are marked as publicly favourite places of all users in the user community simulating common location preferences of a group of users. In another step, other locations are appointed as privately favourite places of each single user representing personal preferences. The simulated paths for each user are composed by selecting random locations where the favourite places (private and public) have a higher probability to occur in those paths. These randomly created paths are then rearranged according to their distance to the former point in order to avoid having "senseless" paths. By doing so, each point is followed by a relatively close point on the path, which refers to the *nearest destination first* scheduling policy.

Using the simulated personalized path patterns created by the *Crowd-sourced Path Simulation Algorithm*, the quantity and quality of the nearest neighbours computed by the location-based collaborative filtering algorithm is increased. A high number of similar users are measured due to the fact that a lot of users share publicly favourite places on their path and also several privately favourite places. Furthermore, through the *nearest destination first* scheduling policy, paths are formed regularly, which minimizes the probability of the case that two paths including similar locations have dissimilar sequences.

There are approximately 1000 paths in the database with different lengths and with users having different number of paths. These paths build the basis for the evaluation, which is done by a server-side script computing the precision and running time of the path matching algorithms.

4.2 Methodology

The following methodology is applied for the evaluation:

1. Perform location-based collaborative filtering, build a nearest neighbour list for each user.
2. Select a user and let him be the current user.
3. Create a set of candidate paths from the historical paths of the user and his neighbours.
4. Select one path from the user's historical path records in the database.
5. Assume the length of this path is n. Take its first $n-1$ points and build a new path with these points. Appoint the new path as the object path.
6. Perform path matching approaches with the object path and the set of candidate paths.
7. Compare the recommendation result produced in step 6 with the nth point on the path processed in step 5. If the result indicates the same functional category of locations, the recommendation is correct, otherwise it is wrong.
8. Accumulate the number of correct recommendations and the total number.
9. Return to step 4 until all paths have been selected once.
10. Return to step 2 until all users have been selected once.
11. Report the final precision and running time.

The precision is defined as the result of dividing the number of correct recommendations by the total number of recommendations, i.e.,

$$precision = \frac{|correct\ recommendations|}{|recommendations|} \tag{14}$$

The hardware and software configuration used during the evaluation is: AMD Phenom II X2 560 3.20 GHz CPU, 3GB RAM, Microsoft Windows XP Professional with Service Pack 3, Apache HTTP Server 2.2.17, PHP 5.3.6 and MySQL 5.5.10.

The evaluation function is performed with each path matching approach alternately in five groups of work load including the quantity of 135, 288, 431, 581 and 696 recommendation tasks. The precision and running time of each group of work load is recorded and illustrated in the form of line charts.

4.3 Results

Figure 4 illustrates the precision of the path matching approaches. *AM* denotes *adjacency matrix matching* algorithm, whereas *MD* stands for *minimum distance matching*. *HA* is the hybrid approach through which recommendations are the logical conjunction of the results produced by the two path matching approaches.

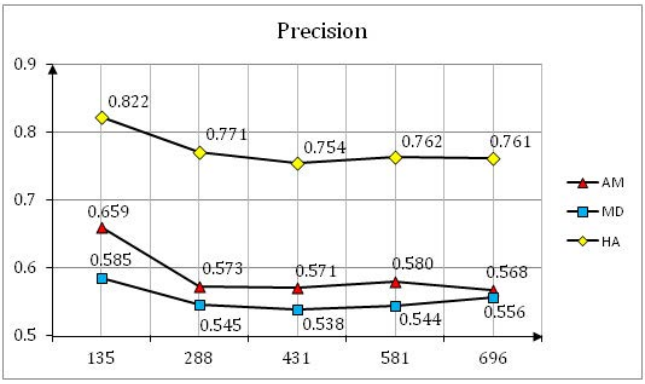


Fig. 4. Path Matching Algorithms - Precision Comparison

The chart shows that *AM* exceeds *MD* in precision. The reason is that *AM* is applied among the paths, which share a majority of path segments (connections between locations) with each other, whereas *MD* is performed even among the paths, which have no shared location. Therefore, the correlation between the paths of *AM* is stronger than that of *MD*. Nevertheless, the performance of the two approaches is not satisfactory enough when being applied individually. In addition, it is also found that when the current path is too short, e.g., containing one or two locations, the *AM* cannot work, because it needs at least two points

on a path to build matrices and an extra point to make a prediction. *MD*, on the other hand, works as usual in that case. Combining both approaches in one recommendation procedure in a hybrid approach, the precision can be significantly enhanced and the drawbacks can be overcome, which indicates that the two approaches complement each other very well. In other words, when *AM* does not perform acceptably caused by lacking shared path segments between two paths, *MD* could replace it for continuous services, and vice versa.

In Figure 5, the running time of the path matching approaches are presented.

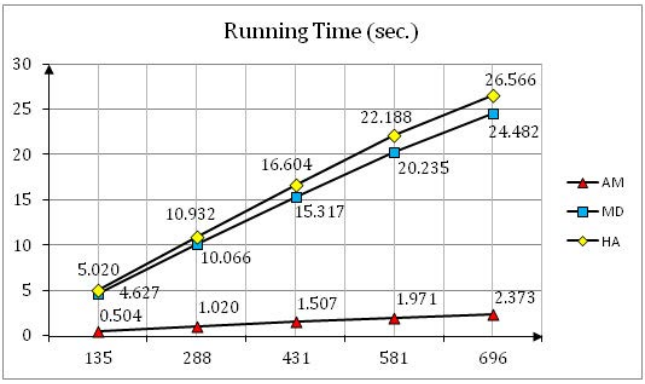


Fig. 5. Path Matching Algorithms - Running Time Comparison

The *AM* approach also performs better than *MD* in efficiency, due to the fact that the *AM* algorithm has lower time complexity of $O(n)$, while the *MD* algorithm with two nested loops has higher time complexity of $O(n^2)$, where n denotes the length of paths. Furthermore, *AM* measures similarity with logical operations, whereas *MD* computes geographical distance with latitudes, longitudes, and trigonometric functions. The running time of the hybrid approach appears to be lesser than the sum of the time of the two single approaches. Considering its precision, the hybrid approach can be regarded as an efficient solution.

The results of testing the recommendation approaches showed that the personalized path patterns produced by the *Crowd-sourced Path Simulation Algorithm* have enhanced the performance of *Tracommender*. The precision of the recommendation approaches computed with personalized path patterns expressed a little superiority over the one computed with random path patterns. Even though the personalized simulating algorithm can be more optimized, we will focus on utilizing personal path information from real users in a future evaluation.

5 Conclusion

In this paper, we proposed a location-based recommender system called *Tracommender* that exploits background tracking data in order to generate location

recommendations. The evaluation shows the feasibility of the concept and very promising results regarding precision and performance. However, more evaluation needs to be performed, especially on real world background tracking data from actual smartphones. In addition, more contextual information can be considered and classified in order to recognize more complex dependencies for improved recommendations.

References

1. Baltrunas, L., Kaminskas, M., Ricci, F., Rokach, L., Shapira, B., Luke, K.-H.: Best Usage Context Prediction for Music Tracks. In: Proceedings of the 2nd Workshop on Context-Aware Recommender Systems, Barcelona, Spain (2010)
2. Bareth, U., Küpper, A.: Energy-Efficient Position Tracking in Proactive Location-based Services for Smartphone Environments. In: Proceedings of the IEEE 35th Annual Computer Software and Applications Conference, Munich, Germany, pp. 516–521. IEEE (2011)
3. Lam, X.N., Vu, T., Le, T.D., Duong, A.D.: Addressing Cold-Start Problem in Recommendation Systems. In: Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication, pp. 208–211. ACM, New York (2008)
4. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender Systems - An Introduction. Cambridge University Press (2010)
5. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-Based Collaborative Filtering Recommendation Algorithms. In: Proceedings of the 10th International Conference on World Wide Web, pp. 285–295. ACM (2001)
6. Baltrunas, L., Ricci, F.: Context-Dependent Items Generation in Collaborative Filtering. In: Proceedings of the Workshop on Context-Aware Recommender Systems, New York, USA (2009)
7. Domingues, M.A., Jorge, A.M., Soares, C.: Using Contextual Information as Virtual Items on Top-N Recommender Systems. In: Proceedings of the Workshop on Context-Aware Recommender Systems, New York, USA (2009)
8. De Carolis, B., Mazzotta, I., Novielli, N., Silvestri, V.: Using Common Sense in Providing Personalized Recommendations in the Tourism Domain. In: Proceedings of the Workshop on Context-Aware Recommender Systems, New York, USA (2009)
9. Takeuchi, Y., Sugimoto, M.: CityVoyager: An Outdoor Recommendation System Based on User Location History. In: Ma, J., Jin, H., Yang, L.T., Tsai, J.J.-P. (eds.) UIC 2006. LNCS, vol. 4159, pp. 625–636. Springer, Heidelberg (2006)
10. Simcock, T., Hillenbrand, S.P., Thomas, B.H.: Developing a Location Based Tourist Guide Application. In: Proceedings of the Australasian Information Security Workshop Conference on ACSW Frontiers 2003, Darlinghurst, Australia, vol. 21, pp. 177–183. Australian Computer Society, Inc. (2003)
11. Fano, A.E.: Shopper's Eye: Using Location-based Filtering for a Shopping Agent in the Physical World. In: Proceedings of the 2nd International Conference on Autonomous Agents, pp. 416–421. ACM, New York (1998)