2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks

Recommend Touring Routes to Travelers according to Their Sequential Wandering Behaviours

Dirksen Liu and Maiga Chang, *Member, IEEE*
School of Computing and Information Systems, Athabasca University, Canada
dirksen@gmail.com, maiga@ms2.hinet.net

Abstract

This paper presents a novel problem of route recommendation which guides the user through a series of locations. The recommendation is made by matching the user's current route with the set of popular route patterns. Sequential pattern mining methods are used to extract the popular route patterns from a large set of historical route records from previous users. An application with real requirements – the Bar Tour Guide – is used to demonstrate the effectiveness of the proposed solution.

Keyword: sequential, route pattern, data mining, mobile, casual wandering

1. Introduction

Planning a route between two any given locations is the main feature of many navigational applications, such as the ever-popular portable/vehicle GPS navigational devices. These applications take route planning as a classical problem of finding the shortest path between two vertexes on a graph. There are many shortest-path finding algorithms, such as Dijkstra's algorithm [3], and its variants A* search algorithm [6], D* search algorithm [13], etc. which can produce route plans efficiently. However, not every case of traveling involves only one destination. This research looks into a different kind of traveling behavior that involves multiple destinations – casual wandering.

This research is motivated by the need to make recommendations for casual wanderers, who travels between a series of locations attracting to them. Examples of casual wandering behavior can be found in small touring activities such as museum going, gallery visiting, zoo exploring, or as big as an excursion in a national park, or a day trip in a city tour. Travelers in these cases are more concerned about visiting the right places – the places they like. On the other hand, they care less about how to minimize the time or distance of traveling, although reasonable economy is expected, such as no wasteful traveling of zigzagging.

The basic strategy of making route recommendation is to extract popular route patterns from a large route database, and recommend the route patterns whose prefixes are most similar to the active user's current route. The underlying assumption of this strategy is that those who agreed in the past tend to agree again in the future, so if the active user's route is similar to the prefix of a popular route pattern, then there's a good chance the active user will follow the rest of the popular route pattern.

The goal of this research is to (1) identify an approach to extract popular route patterns from a historical route database, and (2) to devise a method of finding the top-N recommendations via popular route pattern most similar to the active user's route, effectively and efficiently.

2. Recommender System in Sequential Context

Making recommendations is also the main feature of a large variety of applications called recommender systems, which are achieving widespread success in E-Commerce nowadays. Examples of such applications include recommending books, CDs, and other products at Amazon.com [9], movies by MovieLens [10], and news at VERSIFI Technologies [2].

Recommender systems are usually classified into two categories, based on how recommendations are made [1]: content-based recommendations, and collaborative recommendations. Content-based systems recommend items similar to those that a user liked in the past [7], while collaborative recommender systems (or collaborative filtering systems) try to predict the preference of items for a particular user based on the items previously rated by other users [4]. Typical examples include the book recommendation

978-0-7695-3908-9/09 $26.00 © 2009 IEEE
DOI 10.1109/I-SPAN.2009.98

system from Amazon, the PHOAKS system that helps people find relevant information on the WWW [14] and the Jester system that recommends jokes [5].

The method being proposed to solve the route recommendation problem is inspired by the collaborative methods, in that it also tries to identify the previous users that share the same choice pattern with the active user, and then recommend the next choice made by those previous users to the active user. However, most of the current collaborative recommender systems do not consider the order of choices made by the users, while sequential order is a crucial factor in the route recommendation problem. This issue will be discussed in details in Section 3.

There are a number of recommender systems that take the order of user's choices into account. Shani, Brafman and Heckerman (2005) view the recommendation process as a sequential decision problem and propose using Markov decision processes (a well-known stochastic technique for modeling sequential decisions) for generating recommendations [12]. Tseng and Lin (2006) use n-gram (another derivative from the Markov chain model) based sequential pattern mining techniques to mind the mobile user's web usage sequence, aligned with the location sequence where the user uses the web [15]. Although their objective is to predict the next user request and the next location, to reduce mobile web surfing latency, their work is strongly related the recommender system. After all, making recommendations is to predict what they user likes.

3. Scenario of Bar Tour Guide

The motivation of this research is to find a solution for the Bar Tour Guide application. The main objective of the application is to recommend bar tour routes to its users. A typical scenario of the Bar Tour Guide application usage is as follows. The user carries the Bar Tour Guide handheld device as setting out on a tour. The device constantly monitors the user's routing activities. The user will find the first bar by himself/herself (the application will not make recommendation until the user has made the first choice; more discussion later in the chapter). As soon as the user finishes the drinking and is stepping out of the first bar, the Bar Tour Guide will start the calculation, and presents a list of top-N next popular bar that are most likely favored by the user. The user may or may not take the recommendation, so the next time the user issues a new request for recommendation, the system will recalculate the recommendations based on the most up-to-date user route.

For example, Maz is bar touring in downtown Toronto (see Figure 1 for a map of some bars and pubs in that area, and Maz's route on the map). He first visits *G*, then *A*, and now he wonders which one to go next. So he pulls out his GPS iPhone, which is running the Bar Tour Guide program. The system scans a list of popular bar tour routes, and discovers that many previous bar tour goers, who drank at *G* then *A*, would pick *E* as the next hop. Therefore, *E* becomes the recommendation for Maz.

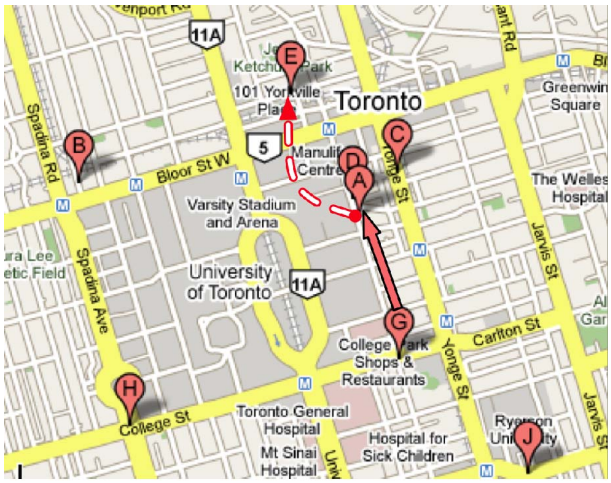

Figure 1. Bar touring in downtown Toronto.

The problem as how to get from one stop to the next in the route on the street level, can be addressed with conventional navigational tools, and thus is out of the scope of this paper.

4. Issues Needed to Solve

First, let us define what a route is. Since we are mostly concerned about the choices made by the bar tour goers, so a route can be represented as an ordered list of locations. We are also only interested in a number of specific types of locations, referred to as significant locations (in this case, drinking establishments). In other words, if the user drops into a café, a flower shop, during his/her bar tour, those stops will not be recorded in his/her route.

Formally, significant locations Λ is a set of predefined geo positions of significant interest, such as the geo positions of all bars in a city in the case of bar touring. The member of Λ is denoted as a 3-tuple *(latitude, longitude, name)*. A user route $route_{userID}$ is an ordered list of locations, denoted by the form $<\lambda_1 \lambda_2 \ldots \lambda_n>$, where $\lambda_j \subset \Lambda$. A sequence with length k is

referred to as a *k-sequence*.

The objective of any recommendation problem is to predict what the user likes. The route recommendation problem, presented by the Bar Tour Guide application, is unique from other conventional recommender system, in that it needs to predict not only what the user likes, but also in what order.

Many recommender systems, especially those that follow the collaborative approach, exploit the observation that like-minded people behave similarly, and thus often make similar decisions again. The basic strategy of the solution being proposed follows the same line of thinking. It takes the following three steps:

1. Look up recurring behavior patterns as what the previous users have chosen to visit, and in what order.
2. Compare the current active user's behavior against the patterns discovered in step one, identify those patterns that matches the current user. Obviously the previous users exhibited such behavior patterns share common preference profiles with the active user, hence what locations they visited next can be presented as recommendations as those locations would be very likely preferred by the active user as well.
3. There is often more than one recommendation. To avoid bombarding the active user with too many options, the system should rank each recommendation according to some certain scheme, to help the user to decide which one to take.

Given a database Θ of routes, and an active user route $route_{userID}$, and according to the three-step strategy, the route recommendation problem can be divided into three research issues:

1. how to extract popular route patterns from the database Θ;
2. how to generate recommendations from the popular route patterns; and,
3. how to rank the relevancy of the recommendations to the active user given his/her route routeuserID?

5. Sequential Routing Patterns

Most current recommender systems consider patterns as sets of commonly chosen items. The order of the items in a set is deemed to be irrelevant. Non-sequential pattern based recommender systems are very popular in product recommendation, such as books, music, movies, etc. In these cases, the involved effort to acquire such products makes little influence on the user's decision making, and thus often ignored.

The case is different for route recommendation. There is a cost of traveling incurred on the users moving from one location to the next. This cost will add up as the users travel through a number of locations, and thus the total traveling cost is determined by the visiting order of these locations. As such, users are no longer making decisions solely based on what they like, but also the cost involved in traveling as well. More precisely, the thinking process now becomes:

1. What do I like to visit next?
2. Is it too far? Do I like it so much that I'm willing to travel that far?
3. Is it in my general travel direction? (to keep the total cost down, a common strategy is to move in a constant direction)

Without considering the sequential order, the pattern will not be able to capture the decision making process behind the user's behavior, and thus will failed to find the user thoughts, and present recommendations that's out of context. In summary, route patterns must be sequential. Given routes are sequences, thus route patterns should be common sub-sequences of the original routes.

What is sub-sequence? Shani, Brafman and Heckerman (2005) and Tseng and Lin (2006) both seek to extract correlation between users by examining the sub-sequences of the users' action sequences in specifc problems [12][15]. Their methods are both based on Markov chain models, more specifically the n-gram model, which is a type of probabilistic model for predicting the next item in a sequence. An n-gram is a sub-sequence of n items from a given sequence. Formally, a sequence $S_i' = <s_{i1}' \; s_{i2}' \; ... \; s_{in}'>$ is said to be a *sub-sequence* of an original sequence $S_i = <s_{i1} \; s_{i2} \; ... \; s_{im}>$, where $n \leq m$, if there exists a strictly increasing sequence of indices, namely $j_1 < j_2 < ... < j_n$, such that $s_{i1}' = s_{ij_1}$, $s_{i2}' = s_{ij_2}$, …, $s_{in}' = s_{ij_n}$.

Sub-sequence is an exact fragment of the original sequence. However, focusing only on local fragments, we would loose sight of patterns that span non-consecutively across the original sequences. For example, $<A \; B \; C>$ and $<A \; D \; C>$ are routes that do not share any consecutive sub-sequences, and thus will be dismissed by any Markov chain model as not having any connections between them. And yet that's not true, as they both contain *A* and *C*, and in the same order. That indicates there is some connection between them. The two users who generate these two routes share something in common. In this case, $<A \; C>$ is a

common pattern occurred in both routes.

Sequential patterns that occur non-consecutively in original sequences are better media to capture the common features of those sequences. Figure 2 demonstrates the power of non-consecutive sequential patterns. It depicts two user routes. User 1 starts from *A*, and after visiting an arbitrary number of locations, *D, E,* and *F*, user 1 arrives *G*. User 2 starts from *H*, and then *D, E, F, I* consecutively. Obviously their routes have a common sequence fragment <*D E F*>. In the example, there are two routes sharing a common fragment <*D E F*>. Since equal number of users visit <*D E F*>, but end up in different destination, the Markov chain model cannot determine the probability difference between the two routes. If we expand our visual scope to the entire bodies of both routes, we can then discover the correlations between *H* and *I*, *A* and *G*.

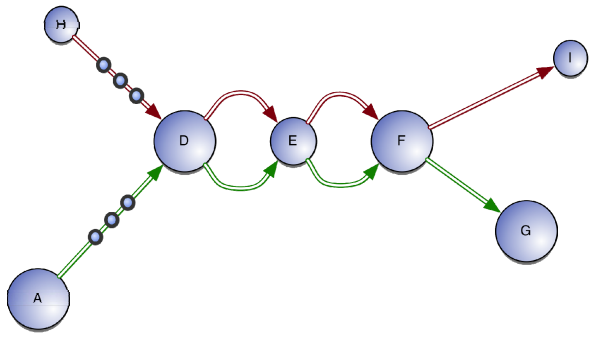

Figure 2. Sequence fragments vs. sequential patterns

The relaxation of the consecutiveness constraint gives rise to a new issue. Suppose *A* and *G* are two locations so far apart that no users who are currently visiting *A*, would make *G* as their next destination. Here a route $route_{userID}$=<*A .... G*> starts from *A*, passes a number of other locations, and finally ends at *G*. By definition, <*A G*> is a sub-sequence of $route_{userID}$, and thus has a chance of becoming a route pattern. This is in contrast to the fact that no one would travel directly from *A* to *G*, and thus not a faithful reflection of the reality. As such, a constraint on distance between adjacent elements is needed to ensure the route pattern itself is a possible route. The upper limit of the distance between adjacent elements of a pattern is a user-defined parameter, referred to as *maximal_distance*.

Summarizing the above discussion, a routing *pattern* can be defined as a recurring sub-sequence with significant frequency in the routing sequence database. The significance of a routing pattern is measured by its *support*. Given a database $\Theta=\{S_1, S_2, ..., S_N\}$ that contains *N* sequences, the *support* of a routing *pattern* is defined as

$$sup(pattern) = \frac{count(pattern \subset S_i \; and \; 1 \le i \le N)}{N}$$

A routing pattern is said to be *popular,* or *frequent*, if its frequency of occurrences in the database is no less than a certain user-defined minimum support (or referred to as the *min_sup*). For example, in a database which has the following three routes: <*A B C D*>, <*E B C F*>, <*H I J*>, with minimum support of 50%, the route pattern <*B*>, <*C*>, <*B C*> are all popular since they appear two times out of 3. A routing pattern is said to be *maximal*, if it is not a sub-sequence of any other popular routing patterns. In the above example, <*B C*> are maximal, while <*B*>, <*C*> are not.

6. Recommendation Rules

After we find the patterns, we need to connect the active user with previous users through route pattern matching and generate route recommendations from the matching route patterns.

The solution to the first task is straightforward. Since a number of previous users are said to be like-minded if their routes contain the same route pattern, then in the same line of thought, an active user whose route also contains this pattern will have a strong connection with these users as well. When a route pattern is found contained in an active user route, it is said to a match to the active user route.

With a number of previous user found to be like-minded to the active user, the next step is to find out suitable locations for recommendations among those visited by this group of previous users, while not yet visited by the active user. More importantly, the suitableness of a location follows these criteria:

1. The location should be visited by a significant number (over min_sup) of users in the previous user group identified by the pattern; if not, it is probably not worth to recommend.
2. The location should occur after the pattern in the previous user route, otherwise the system will recommend the active user to go backward.
3. The location should not be too far (less than maximal_distance) from the last location of the pattern, since the active user would not take it if it were.

If such a location can be found, then appending this new location to the pattern will produce a new pattern, as it satisfy all requirements of route patterns, that it is a popular sub-sequence occurs in a significant number of existing routes, and the distance between every

adjacent elements of which does not exceed *maximal_distance*. Being a route pattern, it will be discovered by the sequential pattern mining process. Therefore we can exploit the route patterns to generate recommendations by converting them into recommendation rules. For a popular route pattern $<s_{i1}\ s_{i2}\ ...\ s_{im}>$, a recommendation rule can be generated as follows:

$rule = <s_{i1}\ s_{i2}\ ...\ s_{i(m-1)}> \rightarrow s_{im}$

By the definition of *rule*, the antecedent $<s_{i1}\ s_{i2}\ ...\ s_{i(m-1)}>$ is also termed as the left hand side – *LHS(rule)*, which is to be used as the pattern to match active user route with; and the consequent s_{im} as the right hand side – *RHS(rule)*, which is the recommendation to present to the user.

Given an active user route $route_{userID}=<s_{i1}\ s_{i2}\ ...\ s_{in}>$, the *rule* is said to match or applicable to $route_{userID}$ if $origin(rule) \subseteq route_{userID}$ (*origin* is a function that returns the original route pattern for any given *rule*), and $s_{i(m-1)}=s_{in}$. The additional constraint $s_{i(m-1)}=s_{in}$ is necessary to avoid making out-of-context recommendations. For example, *A, B, C, D* are four locations on a street, with *A* on one end, and *D* on the other. A user has visited *A, C, D*. Without the additional constraint, his route $<A\ C\ D>$ matches the rule $<A>\rightarrow<B>$, therefore the system will suggest *B* as the next location, which makes no sense to the user, as the user has clearly passed *B*.

As sequential mining methods will return all popular patterns, popular 1-sequences (sequences of length 1) will be among the discovered pattern as well. However, the above definition excludes popular 1-sequence from recommendation rules generation. If the only one item in the 1-sequence is used as recommendation, then there leaves nothing for the system to infer connections between the active user and the popular 1-sequences, and thus the system will be unable to make any recommendation at all.

This is well known issue called *cold start*, common among recommender systems. A simple solution is to recommend the locations in those popular 1-sequences which are within the *maximal_distance* radius to the active user's current location. If none can be found, then the system should refrain from making any recommendations.

For an active user, there could be a number of matching route patterns, leading to multiple recommendations. This is most common at the beginning of the tour. To avoid bombarding the user with too many options, the system should rank the recommendations to help user to choose. In other words, the purpose of the ranking is to estimate how well the active user would like each of the recommendation, which is in turn determined by how strong a connection exists between the rule that generates this recommendation and the active user.

Due to the page limitation, we can not talk the solution of the bombarding recommendation issue further in details. We use the strength concept and the rule similarity to measure the recommendability of a $rule_i$.

7. Conclusions

This research studies a novel problem – route recommendation based on behavior patterns. Based on the observation that user's routing behavior is a sequential decision making process, the concept of sequential patterns is being used to define behavior patterns. And as a result, sequential pattern mining methods are used to extract popular behavior patterns, which are then turned into a set of recommendation rules. Given an active user, the system will first find out which rules are applicable to the user, sort the rules according to a ranking scheme, and finally present the top-n highest-ranking rules' *RHS* as the recommendations to the user. We will do the simulation test to verify the effectiveness of our method. Furthermore, build the application on the mobile phone and do the pilot experiment is also our next step.

We have designed three major algorithms to make the research objectives come true, the three algorithms are GPSToRoutingSequence, GSP-with-maximal-distance-constraint, and FindRecommendation. We have also simulated experiment by ourselves along the College Street in downtown Toronto. The *maximal_distance* we set in the experiment is 3 meters. Also, we use t_{gap} as 30 minutes, i.e. the user has to stay in one location for at least 30 minutes so that the location is qualified as an element in the user's route. Due to the experiments were simulated by us, we can't use a simple questionnaire to verify if the recommendation is fit the user requirements. We have a basic definition about the recommendation, i.e. "recommendation system should only provide users suggestions rather than force them to comply". The mobile recommendation mechanism and system we designed always give the users multiple choices and make them decide, we simply record the user routing behaviors and give them the next options.

Sequential pattern mining is the pivotal process in the solutions to this problem. Currently GSP is being employed to mine sequential patterns. However, with

its inheriting limitation, GSP does not scale well to meet any realistic demand, such as the demand from the Bar Tour Guide application. Therefore, an urgent need is required to adopt a more advanced sequential mining method for behavior pattern extraction. Given the incremental nature of the routes database used in the Bar Tour Guide, incremental sequential pattern mining methods, such as the ISM algorithm [11], and the IncSP algorithm [8], are the most helpful candidate replacements. Therefore, future research should look into how to extend existing incremental sequential pattern mining methods to honor the *maximal_distance* constraint, and apply them into the behavior pattern extraction process.

References

[1] Balabanovic M. & Shoham, Y. (1997). Fab: Content-Based, Collaborative Recommendation. Communications of the ACM, 40(3), 66 – 72.

[2] Billsus, D., Brunk, C.A., Evans, C., Gladish, B. & Pazzani, M. (2002) Adaptive Interfaces for Ubiquitous Web Access. Communications of the ACM, 45(5), 34 – 38.

[3] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1, 269 – 271.

[4] Goldberg, D., Nichols, D., Oki, B.M., & Terry, D. (1992). Using Collaborative Filtering to Weave an Information Tapestry. Communications of the ACM, 35(12), 61 – 70.

[5] Goldberg, K., Roeder, T., Gupta, D. & Perkins, C. (2001). Eigentaste: A Constant Time Collaborative Filtering Algorithm. Information Retrieval, 4(2), 133 – 151.

[6] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100 – 107.

[7] Lang, K. (1995). Newsweeder: Learning to Filter Netnews. In the Proceedings of the 12th International Machine Learning Conference, (ML 1995), July 9, 1995, Tahoe City, California, USA, 331 – 339.

[8] Lin, M.-Y., & Lee, S.-Y. (2004). Incremental Update on Sequential Patterns in Large Databases by Implicit Merging and Efficient Counting. Information Systems, 29(5), 385-404.

[9] Linden, G., Smith, B., & York, J. (2003). Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing, 7(1), 76 – 80.

[10] Miller, B.N., Albert, I., Lam, S.K., Konstan, J.A. & Riedl J. (2003). MovieLens Unplugged: Experiences with an Occasionally Connected Recommender System. In the Proceedings of the 8th International Conference on Intelligent User Interfaces, (IUI 2003), January 12 - 15, 2003, Miami, Florida, USA, 263 – 266.

[11] Parthasarathy, S., Zaki, M. J., Ogihara, M., & Dwarkadas, S. (1999). Incremental and interactive sequence mining. In the Proceedings of the 8th International Conference on Information and Knowledge Management, (CIKM 1999), November 2-6, 1999, Kansas City, Missouri, USA, 251 – 258.

[12] Shani, G., Brafman, R. I., & Heckerman, D. (2005). An MDP-Based Recommender System. Journal of Machine Learning Research, 6(2), 1265 – 1295.

[13] Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In the Proceedings of the 1994 IEEE International Conference on Robotics and Automation, (ICRA 1994), 8-13 May 1994, San Diego, CA, USA, 3310-3317.

[14] Terveen, L., Hill, W., Amento, B., McDonald, D. & Creter, J. (1997). PHOAKS: A System for Sharing Recommendations. Communications of the ACM, 40(3), 59 – 62.

[15] Tseng, V. T., & Lin, K. W. (2006). Efficient mining and prediction of user behavior patterns in mobile web systems. Information and Software Technology, 48(6), 357 – 369.