RecTour: A Recommender System for Tourists

Ranieri Baraglia*, Claudio Frattari†, Cristina Ioana Muntean‡
Franco Maria Nardini*, Fabrizio Silvestri*
*ISTI–CNR, Pisa, Italy
{name.surname}@isti.cnr.it
†University of Pisa, Pisa, Italy
claudiofrat2002@hotmail.it
‡Babes-Bolyai University, Cluj-Napoca, Romania
cristina.muntean@econ.ubbcluj.ro

***Abstract*—This paper presents a recommender system that provides personalized information about locations of potential interest to a tourist. The system generates suggestions, consisting of touristic places, according to the current position and history data describing the tourist movements. For the selection of tourist sites, the system uses a set of points of interest a priori identified. We evaluate our system on two datasets: a real and a synthetic one, both storing trajectories describing previous movements of tourists. The proposed solution has high applicability and the results show that the solution is both efficient and viable.**

***Keywords*-tourist recommender systems**

I. Introduction

The growing popularity of portable computing devices such as smart-phones, tablets, etc., has promoted the digital medium to be used in any context and with every opportunity. The increasing amount of information available on the Web leads to the need of finding new solutions that ease the availability and enhance the use of such information. Recommender systems fit into this scenario, and deal with providing personalized information to users according to their interests. Such systems are have a high applicability in situations where a focused suggestion brings value to users.

This work presents a recommender system that offers personalized information about locations of potential interest to a tourist. The system generates suggestions consisting of touristic places, according to the current position (and history) of a tourist that is visiting a city. For the selection of tourist sites, the system uses a set of points of interest (PoI) identified a priori, that may relate to monuments, restaurants, hotels, information center, etc. For computing suggestions, the system uses knowledge gained from the analysis of a set of trajectories describing the routes previously followed by tourists.

In the past several recommender systems using trajectory-based models as well as the concept of PoI have been proposed. In [9], authors perform two types of travel recommendations by mining multiple GPS traces. The first is a generic one that recommends top interesting locations and travel sequences in a given geospatial region. The second is a personalized recommender that provides a user with locations matching her travel preferences. Monreale *et al.* propose "WhereNext", a method aimed at predicting with a certain accuracy the next location of a moving object [6]. The prediction uses previously extracted movement patterns named *Trajectory Patterns* which are a concise representation of behaviors of moving objects as sequences of regions frequently visited with typical travel time. A decision tree, named T-pattern Tree, is built from the Trajectory Patterns and evaluated with a formal "training and test" process. The tree may be used as predictor of the next location of a trajectory, finding on the tree the best matching path. In [8] a solution exploiting item-based techniques to suggest new PoIs to users is proposed. The system recommends shops based on users' past location data history. Shops frequented by the users are used as input for the item-based collaborative filtering algorithm that makes recommendations. Points of Interests are also used in combination with collaborative filtering techniques [1]. Here, authors propose a mobility-aware recommendation system that uses the location of the user to filter recommended links. In [5] the proposed algorithm interactively generates personalized recommendations of touristic places based on the knowledge mined from photo albums and Wikipedia. Tourist recommendations are computed using random walks.

II. The proposed system

The proposed recommender system has two main components: offline and online. The first one aims to create the knowledge model, which is the basis for computing suggestions. Its execution takes place when new data is available for updating the knowledge model, and is based on the trajectory dataset and the PoIs. The online component uses the current user information and the knowledge model in order to produce a list of suggestions.

Building the Knowledge Model. The data processed by the offline component consists of a dataset of trajectories representing the movements of users in a certain period of time, as detected by their GPS devices, and a set of PoIs including their coordinates. The trajectories initially have the following format: $T =< (x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_n, y_n, t_n) >$, where (x, y) are the coordinates on the Cartesian plane, and t is the timestamp.

To facilitate the process of mining trajectories we define the following distance function: $N : \Re^2 \rightarrow P(\Re^2)$, describing whether the points of a trajectory are related to a PoI. In order to do this, the Cartesian plane is divided into regions

so that if a point falls in the region of a PoI, then this point is assigned to that region. This function allows us to simplify the process of mining, which manages trajectories expressed as: $T'' =< (A, t_1), (B., t_2), ..., (C, t_n) >$, where capital letters represent regions and t timestamps, instead of trajectories expressed as coordinates. It reduces the cardinality of the set of elements on which the knowledge model is computed. Choosing the appropriate strategy to divide the plane into regions is not a trivial problem. We have to take into account several factors such as the PoIs distribution on the plane, the granularity of objects to suggest, the adopted knowledge model and the adopted algorithm computational costs. Moreover, the division of the plane can be independent from the number of PoIs or we can have unique pairs of PoI and region, i.e. each region consists of a PoI. We considered three methods: Grid, Voronoi and QuadTree. The Grid [3] divides the plane in a series of contiguous cells of the same shape and size, to which to assign unique identifiers used for spatial indexing. Using this model for our data would mean that every region of the grid can have from 0 to n PoIs. We would also have to consider the correspondence $IDregion \rightarrow POIs$ and the order in which PoIs within the same region should be suggested or the coordinates of PoIs to calculate the distance between regions (i.e. cells). The grid-based method is useful if we want to operate at different scales (i.e., city, area, neighborhood), but for searching and comparison of the elements, tree structures (e.g., QuadTree and R-tree) behave better.

Voronoi tessellation (Vot) [2] is a special kind of decomposition of a metric space determined by distances to a specified discrete set of points in the space. Given a set of PoIs, a Vot containing one PoI per area can be computed. As a result we obtain a space divided into several areas, each one with a different shape and size. The points contained within a PoI's area have the characteristic of being closer to that area than any other PoI. To assign these areas to the trajectories' points causes trajectories to describe paths of movement from one PoI to another. If there is a too fine subdivision of the space, this can lead to a loss of accuracy also due to the tolerance of GPS devices. On the other hand if we have very large areas including many PoIs, this can lead to make equal two trajectories that in practice are different. To overcome this problem, clusters of PoIs can be built, and the cluster midpoint could be exploited to make the Voronoi tessellation form macro-areas with a homogeneous density. A matching among PoIs and clusters is also required. Since Vot makes irregular shape regions, it makes difficult to assign trajectory points to regions. Like the Grid, Vot is not a hierarchical structure and suffers for the same disadvantages for searching and comparing corresponding elements. To this end we choose to adopt the QuadTree technique [7].

A QuadTree-based algorithm divides the plane into regions of the same shape, but of different sizes. The regions can be very small where there are several close PoIs. Their size depends on the PoIs distribution, and especially on the distance from one PoI to another. The QuadtTree technique is very efficient, it performs data comparison and data insertion with a complexity equal to $O(logn)$, where n are the number of PoIs in input. In the worse cases, corresponding to poor PoIs distributions and thus unbalanced trees, it can run in $O(n)$.

In our case we have an initial set of PoIs, and it is reasonable to imagine that in the central areas there are more POIs than in peripheral ones. In addition, the PoIs were divided according to the following categories: tourist attractions, places to eat, places to stay, tourist points, mobility points and aid points. We take into consideration that a tourist may not be interested in all types of PoIs, thus we can choose to make a division of the plane only for a specific category.

For constructing the knowledge model we first transform the dataset of trajectories of regions according to the QuadTree division of space. This allows us to identify points that fall in a specific region with identifier ID. Thus the trajectories are represented as: $A \underset{\alpha}{\rightarrow} B$, where A and B are two regions and α is the estimated time needed to move from A to B. Then, to build the knowledge model, the T-Pattern Tree [6] is used. Frequent trajectories are trajectories with a support greater than a threshold value σ. The support refers to the threshold that defines how many times the trajectory should appear in a set to be considered frequent. The T-pattern Tree is built incrementally and trajectories having a prefix in common are overlapped on the tree to avoid unnecessary branches duplication. Each node is identified by a tuple $< id, region, support >$, where id is the node identifier, $region$ is the concerned region and $support$ is the sum of the supports of the various trajectories that have the $region$ in that position. Every path on the tree represents a frequent trajectory as a sequence of steps from one region to another. Each edge has associated a tuple $< [t_{min}, t_{max}], P(A \rightarrow B) >$, where $[t_{min}, t_{max}]$ represents the estimated time needed to pass from a parent region to a child one, and $P(A \rightarrow B)$ indicates the probability of moving from region A to region B. The probability value is computed as the trajectories support value of a child node divided by the trajectories support value of its parent node.

Computing suggestions. User locations are collected by GPS systems and sent to the offline component whenever a new position is detected. The recent movements of a current user over a period of time are used to set up the *current trajectory*, which is then compared with each practicable T-pattern Tree path. From time to time, as a trajectory is compared, a pruning operation takes place to eliminate the patterns with which there is a minimum non-correspondence with the current trajectory. Doing so, the number of comparisons to lengthen the path decreases. For each pattern match a score, called *Punctual Score*, is calculated by assigning a value to each node of the current trajectory, which is then compared with the T-pattern Tree nodes. The comparison may lead to three different cases for the calculation of the Punctual Score: 1) The current region is equal to that of the node, and it is reached within the expected time. The punctual score is equal to the support associated to the node; 2) The current region is equal to the current node, but it is not reached within the maximum expected time. The punctual score is computed as: $^{node.support}/_{\beta * d_t}$, where β is a constant and d_t is the

distance in time between the end interval and the time when the current region is reached; 3) The current region is not equal to that of the current node, the punctual score is computed as: $node.support/(\beta * d_t + \alpha * d_s)$, where α and β are constants, d_t and d_s are respectively the distance in time and in space between the current region and that of the node.

Moreover, the distance time tolerance th_t and distance space tolerance th_s were defined. These thresholds specify the maximum value that d_t and d_s can assume; when they exceed the specified values the Punctual Score of the node is set equal to 0, i.e. the current region is not close to that of the node or it is not reached within the expected time. The total pattern three score $PathScore$ is computed as follows. Given a trajectory tr, a path $P = [p_1, p_2, ..., p_k]$ and a punctual score $PScore_k$ defined on each $p_k \in P$, the three indexes can be computed as:

- $AvgScore(tr, P) = \frac{\sum_1^n PScore_k}{n}$
- $SumScore(tr, P) = \sum_1^n PScore_k$
- $MaxScore(tr, P) = max\{PScore_1, \dots PScore_k\}$

In our tests we use the $SumScore$ method and assign the highest score to the longest path that intersects the trajectory.

To carry out suggestions, the list of candidates and the probability to move from a candidate parent to a candidate child are computed. The candidates with higher $PathScore$ and their children are returned as a suggestion, indicating the next regions that one can get from the current position. Once the regions are found, we look for the associated PoIs, which are suggested to the tourist. We can use different policies to choose which PoIs to suggest. In our study we consider PoIs divided into several categories, which could be handled in two different ways: 1) build a unique knowledge model with all the PoIs; 2) build a knowledge model for each category of PoIs. We opted for a mixed solution. First, we create a knowledge model for the category of tourist sites. The T-pattern Tree describes the flow of movements from one region to another. Then, by default the user will receive suggestions on tourist places that are updated according to his moves. Moreover, by using the information associated to the edges we can also provide, as a function of the available means of transport, an approximate time and cost needed to reach a suggested place. Suggestions for the other categories are not inferred by the knowledge model. A user can issue at any time a request for suggestions for such categories. Suggestions are ranked according to a Euclidean distance function, which computes the distance w.r.t. the last location in the current trajectory.

III. Evaluation

We evaluate the effectiveness and the efficiency of the proposed system by using two trajectory sets: synthetic and real, and a set of predefined PoIs. Moreover, the performance results obtained by the proposed solution were compared with those obtained by a greedy solution that carries out a list of suggestions made up of regions closer to the current location.

The synthetic set was created using a trajectory generator that takes in input $1,022$ tourist PoIs of Florence, which are combined in sequences to form $20,000$ trajectories. PoIs were generated using information from Wikipedia. A trajectory was built by randomly choosing a PoI from which to start. Then, it takes into account as possible destinations only points at a distance not exceeding a threshold d. A value expressing an interest is associated to each destination. As successfully proposed in [4], the interest of a possible destination is modeled proportionally to the distance between the current user position and the location of the destination target. Based on these parameters the next destination is selected.

The real set is made up of $1,650$ daily user trajectories built using data coming from Flickr. The trajectories are constructed using the photos submitted by users from January 2004 to January 2010. A photo may have additional data such as the time it was taken and the geospatial coordinates of the object depicted, considered potential POIs.

The quality of the trajectory set is a key element for the building of the knowledge model. For example, if a set of trajectory patterns covers only a small portion of the total space in which the users travel, there will be many trajectories for which it will not be able to produce useful suggestions. Therefore, it is important to understand in advance whether a set is valid for the effective evaluation of suggestions. To this end, in [6] the authors have proposed a method to establish a correspondence between the accuracy and the value of the support for a set of association rules. In our case, the ability to make accurate predictions also depends on the spatial characteristics of a set of T-pattern, not only on support. We refer to it as *Coverage*.

According to [6] the following indexes were adopted by us to assess the Coverage of a sample:

- **Spatial Coverage** (SP) measures the fraction of the total space covered by the trajectory set as: $SpatialCoverage = \frac{\cup_{Tp \in Tp_{set}} Space(Tp)}{TotalSpace}$ where $Space(Tp)$ is the function that assigns to each T-pattern a portion of the plane that fails to cover. $TotalSpace$ is the total space where tourists move around;
- **Data Coverage** (DC) defines the fraction of trajectories that go over the support value. This is computed as: $DataCoverage = \frac{|T - |Tpset|}{|t|}$ where T is the trajectory set, $Tpset$ is the number of extracted trajectories that satisfy the support value.
- **Region Separation** (RS) measures the prediction accuracy as function of the prediction granularity. It is computed as: $RegionSeparation = \frac{MinimalRegion}{AVG_{r \in Tp \in Tp_{set}}}$ where $MinimalRegion$ is the minimum spatial granularity corresponding to a PoI within the considered space and $AVG_{r \in Tp \in Tp_{set}}$ is the average size of regions belonging to the trajectory set.
- **Rate** correlates all above three metrics as follows: $Rate = SpatialCoverage \cdot DataCoverage \cdot RegionSeparation$

As can be seen from Table I, as the number of PoIs per region increases, the value of the Rate index decreases. Even

if the Spatial Coverage index increases, the other two indexes Region Separation and Data Coverage decrease. In fact, the higher values for Rate are obtained when there are five PoIs per region. This is because RegionSeparation, and consequently Rate, rewards the correspondence between the PoIs and regions. Almost identical Rate values were obtained for the same test with synthetical sets 5000 and 10000 trajectories. It shows that Region Separation is an index independent by the size of the trajectory set.

Dataset	POI Region	SC (%)	RS (%)	DC (%)	Rate
Real	5	0,70	0,12	0,75	0,06
	10	0,67	0,06	0,75	0,03
	20	0,86	0,03	0,69	0,02
	30	0,83	0,02	0,66	0,01
Synthetic	5	0,91	0,92	0,71	0,08
	10	0,96	0,05	0,64	0,03
	20	0,97	0,03	0,61	0,02
	30	1	0,02	0,06	0,01

TABLE I
COVERAGE VALUE FOR THE REAL AND SYNTHETIC TRAJECTORY SETS

To evaluate the effectiveness of suggestions we adopted an empirical approach that estimates the percentage of errors in making recommendation using a "test set" [6]. The set of samples is divided into two disjoint subsets, a "training set" used to build the knowledge model (90%) and a "test set" (10%). Initially the current trajectory is represented by the first region of the analyzed trajectory and the remaining regions are used for comparison. A trajectory is divided in this way until the second part contains a single region. Each time a trajectory of the test set is divided, a new current trajectory is originated, and the recommender system is run. As a result, a set of suggestions for the current trajectory is carried out. Then, the recommendations are compared with the part of the trajectory that is not part of the current trajectory. The tests to evaluate the efficacy of the proposed solution were conducted by computing a list of 10 regions as a suggestion. For evaluating the effectiveness of the proposed system we adopted the following metrics:

- **Prediction Rate** (PR) is the percentage of trajectories for which the system is able to make a prediction.
- **Accuracy** (A) is the percentage of trajectories for which the system returns a list of suggestions containing the region that in the test set immediately follows the last region in the current trajectory.
- **Average Error** (AE) is the average error percentage computed for each trajectory. As already stated, in a test a trajectory is iteratively divided into two parts: the first part is the current trajectory, and the second one is used to make suggestions. Thus a trajectory of n regions is divided $n-1$ times, and $n-1$ comparisons are made. The result of each comparison is true, if the list of suggestions contains the next region, and false otherwise. Let a the number of false comparisons, the error rate for the related trajectory is equal to $a/n-1$. So, the Average Error is computed as the ratio between the error rate and the number of current trajectories that can be generated.

Table II shows the values of the metrics evaluated to measure the efficacy of the computed suggestions. They were computed by varying the number of PoIs per region and using a knowledge model built with the value of the support equal to 1. The number of PoIs in a region significantly affects the effectiveness of the system. Increasing the number of PoIs per region, the regions become larger and the prediction becomes easier because the knowledge model needs fewer examples to correctly predict the regions. Accordingly, the probability that a region is correctly suggested increases. In fact, the best value for Accuracy is achieved with 30 PoIs per region, with a Prediction Rate of 56.49%.

As can be seen from Table II, Average Error decreases as the number of PoIs in the regions increases, leading to a higher accuracy in suggestions. Prediction Rate computed on the synthetic set is greater than the one on the real set.

Dataset	PoIs	PR	A (%)	AE (%)
Real	5	48,24	30,45	74,35
	10	48,24	47,27	63,29
	20	56,47	59,80	48,10
	30	56,49	66,21	39,39
Synthetic	5	100	49,52	46,12
	10	100	47,46	49,40
	20	100	67,55	30,15
	30	100	81,36	16,15

TABLE II
ACCURACY MEASUREMENTS VARYING THE NUMBER OF POIS PER REGION.

Table III shows the values of the performance metrics evaluated by varying the value of the support (σ) used to generate the knowledge model. Increasing the support, the percentage of trajectories correctly predicted decreases, revealing the non monotonic property of the support. Also, the number of trajectories used to build the knowledge model decreases and the model loses part of its predictive power. Even if for small values of the support the Prediction Rate is high enough, the Accuracy never increases above 50%, instead decreases progressively. In addition, as the value of the support increases, the related knowledge model contains less paths, and therefore less trajectories. The support value that ensures the best Accuracy is equal to 1 for both the real and synthetic trajectory sets.

The system efficiency was evaluated measuring the average elapsed time to compute a list of suggestions on a trajectory. This time may depend on several factors; the main one is the cardinality of T-pattern Tree. The bigger and deeper the tree, the more the execution time grows and the cost of prediction rises. It is intuitive to say that average response time decreases as value of the support increases. In Fig. 1 the average elapsed time to execute a recommendation is computed varying the number of requests to the systems. We test the efficiency of RecTour in handling an increasing number of requests and demonstrate the system is able to respond quickly to a large number of requests with an average elapsed time equal to 1500

ms. For 100 requests per minute the average response time remains constant below 500 ms, then slightly increasing in the case of 150 requests per minute. Reasonable values are obtained also in the case of 200 requests per minute, average response time values being reached around 1400 ms.

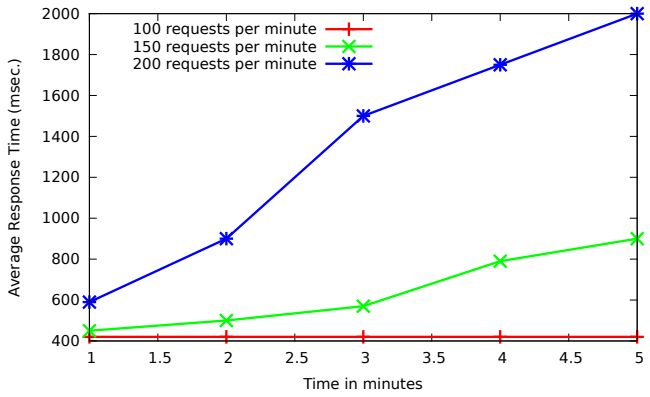

Fig. 1. Average elapsed time to execute a recommendation.

Dataset	σ	PR (%)	A (%)	AE (%)
Real	1	48,24	47,27	63,29
	2	48,24	43,18	64,68
	4	36,18	51,51	63,82
	6	36,18	45,45	71,32
	8	36,18	36,36	76,80
	10	36,18	26,36	63,82
Synthetic	1	100	47,46	49,40
	2	100	42,70	52,63
	4	100	37,76	57,42
	6	100	33,74	61,10
	8	100	30,92	63,75
	10	100	28,80	67,65

TABLE III
TEST RESULTS VARYING THE SUPPORT AND 10 POI PER REGION.

We compare our proposed system with a simple baseline. The baseline consists of a greedy solution which does not adopt any process of mining. Due to the scarcity of recommendation systems available and the low availability of datasets used in other articles, we developed a simple recommender system which returns a list of suggestions containing regions closest to the current region of the tourist. The comparison was made by varying the number of POIs per region and using a support value equal to 1. For the real set Accuracy refers to a value of Prediction Rate smaller than 100%, while Prediction Rate is equal to 100% for the synthetic set and the baseline. This fact is justified by the very small size of the real set, where the training set provides only 405 trajectories for the construction of the knowledge model. To get conclusive results we need to use a fully comparable set. Thus when we compare the baseline to RecTour using the synthetic set, the predictions appear to be more accurate in the case of our recommendation system. They both achieve a Prediction Rate of 100% but we can see a big improvement in Accuracy. For 5 PoIs per region the baseline achieves an Accuracy of 20,38%, while RecTour achieves an Accuracy of 49.52%. By increasing the number of PoIs per region, Accuracy increases as well for both methods. However, at 30 PoIs per region, the baseline has a 60.4% Accuracy and RecTour has an 81.36% Sccuracy, thus our method outperforms the baseline.

IV. Conclusions and Future Work

We designed and implemented RecTour: a recommendation system that can assist a tourist visiting a city. RecTour is able to generate suggestions consisting of potentially interesting Points of Interest, depending on the current position of the tourist himself. We assessed RecTour in terms of efficiency and effectiveness. To do so, we used a real dataset of 1,650 trajectories and a synthetic datasets of 20,000 trajectories. Tests were conducted by varying the value of support and the number of POIs per region. The first leads to different knowledge models while the second to different divisions of the geographical plane. Tests using the synthetic dataset obtained a higher accuracy than that obtained in the case of the real dataset. This is mainly due to the fact that the real data set is smaller thus the percentage of prediction accuracy is affected.

The best effectiveness is achieved when the support is $\sigma = 1$ and when the number of PoIs per region is equal to 30. Given these parameters, the real dataset obtained a Prediction Rate of 56.49%, while the synthetic dataset achieved a Prediction Rate of 100%. We also evaluated our proposed system against a simple baseline, which produces a suggestion list of regions closer to the tourist's current position. Results showed that RecTour clearly outperforms the baseline. Furthermore, we also proved that the response time of RecTour enables it to be used interactively. Based on the results we can conclude that RecTour is a viable and efficient system which can produce useful recommendations for tourists while visiting a city.

Our research has been funded by the POR-FESR 2007-2013 No 63748 (VISITO Tuscany) project and POS-DRU/88/1.5/S/60185 (Investing in people!).

References

[1] M. Brunato and R. Battiti. Pilgrim: A location broker and mobility-aware recommendation system. In *Proc. of PerCom*, pages 265–272. IEEE, 2003.

[2] Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Rev.*, 41(4):637–676, December 1999.

[3] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. In Pavel Berkhin, Rich Caruana, and Xindong Wu, editors, *KDD*, pages 330–339. ACM, 2007.

[4] K. Lee, S. Hong, S.J. Kim, I. Rhee, and S. Chong. Slaw: A new mobility model for human walks. In *Proc. IEEE INFOCOM*, pages 855–863. IEEE, 2009.

[5] C. Lucchese, R. Perego, F. Silvestri, H. Vahabi, and R. Venturini. How random walks can help tourism. In *In Proc. ECIR*. LNCS, 2012.

[6] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *Proc. of KDD*, pages 637–646. ACM, 2009.

[7] H. Samet. Hierarchical spatial data structures. *Design and Implementation of Large Spatial Databases*, pages 191–212, 1990.

[8] Yuichiro Takeuchi and Masanori Sugimoto. Cityvoyager: An outdoor recommendation system based on user location history. In *Ubiquitous Intelligence and Computing*, LNCS. Springer Berlin / Heidelberg, 2006.

[9] Y. Zheng and X. Xie. Learning travel recommendations from user-generated gps traces. *ACM TIST*, 2(1):2, 2011.