Where Shall We Go Today? Planning Touristic Tours with TripBuilder

Igo Brilhante,
Jose Antonio Macedo
Federal University of Cearà, Fortaleza, Brazil
{igobrilhante,jose.macedo}@lia.ufc.br

Franco Maria Nardini,
Raffaele Perego, Chiara Renso
ISTI–CNR, Pisa, Italy
{name.surname}@isti.cnr.it

ABSTRACT

In this paper we propose TripBuilder, a new framework for personalized touristic tour planning. We mine from Flickr the information about the actual itineraries followed by a multitude of different tourists, and we match these itineraries on the touristic Point of Interests available from Wikipedia. The task of planning personalized touristic tours is then modeled as an instance of the Generalized Maximum Coverage problem. Wisdom-of-the-crowds information allows us to derive touristic plans that maximize a measure of interest for the tourist given her preferences and visiting time-budget. Experimental results on three different touristic cities show that our approach is effective and outperforms strong baselines.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—|*Information Filtering, Search process*

Keywords

Tourist Trip Recommendation, Trajectory Mining

1. INTRODUCTION

Planning a travel itinerary is a complex task for tourists approaching their destination for the first time. Different sources of information such as travel guides, maps, on-line institutional sites and travel blogs are consulted in order to devise the right blend of Points of Interest (PoIs) that best covers the subjectively interesting attractions and can be visited within the limited time planned for the travel. Moreover, the tourist has to guess how much time is needed to visit each attraction and to move from one PoI to the next one.

These simple considerations motivate our proposal of TripBuilder, an unsupervised approach that possibly overcomes the above limitations by exploiting the wisdom-of-the-crowds from past tourists to build a personalized plan of visit. TripBuilder takes as input the target destination, the time available for the visit, the (explicit or implicit) user's profile, and builds a personalized tour crossing a selection of the PoIs. The recommended tour maximizes user's interests and respect the visiting time constraint since it takes into account both the time to enjoy the attractions and the time needed for moving from one PoI to the next one. Moreover, the knowledge used to feed the TripBuilder recommendation model is entirely extracted in an unsupervised way from two publicly available collaborative services: Wikipedia and Flickr. Thereinafter, we will consider touristic cities as the destination targets of our users, although our technique is general and scale-independent and could be applied even to build travel itineraries crossing large regions or countries.

More in details, TripBuilder - our novel framework for building trip plans - consists of the following contributions:

- we introduce an unsupervised method for mining common patterns of movements of tourists in a given geographic area. The method uses i) Flickr, to gather public photos (and their metadata) from users all around the world, and ii), Wikipedia to gather information regarding Points of Interest (PoIs) in the given touristic city. The results of our unsupervised method is a touristic database storing PoIs, their popularity, visiting time, categorization, and the patterns of movement of tourists that visited them in the past;
- we define the TripCover problem as an instantiation of the Generalized Maximum Coverage (GMC) problem. We model each visiting pattern by means of the PoIs and the associated Wikipedia categories, and the GMC profit function by considering PoIs popularity and the actual user preferences over the same Wikipedia categories. The cost function is instead built by considering the average visiting time for the PoIs in the patterns plus the time needed to move from one PoI to the next one. Our algorithm is thus able to provide visiting plans made up of actual touristic itineraries that are the most tailored to the specific preferences of the tourist;
- we build three real-world datasets from Wikipedia pages and photos taken in three different cities of large touristic interest. We made the three datasets available for downloading to favor reproducibility of our results and advances in the field;
- we present detailed experiments showing that our solution outperforms strong baselines.

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.
Copyright 2013 ACM 978-1-4503-2263-8/13/10 ...$15.00.
http://dx.doi.org/10.1145/2505515.2505643.

2. RELATED WORK

Several works in the literature are proposing methods to recommend list of PoIs based on the actual location of the user [8, 15, 14, 7]. In this paper, instead of producing a list of candidate PoIs based on the location of the user, we investigate a method for suggesting "touristic trips" made of time-budgeted itineraries followed by actual tourists that cross PoIs matching the user preferences. Even if TRIP-BUILDER could be easily adapted to fit a Location-based Service Model, in this paper we build precomputed itineraries that are recommended to the user **before** the visit, at the planning stage. In this sense our approach can be considered a step further the simple location-based PoI recommendation. The task of designing a trip for a tourist approaching a new city has been investigated in the literature as we can see in the interesting survey on the topic presented in [11].

An early work on this topic is [4]. Authors use the Traveling Salesman Problem (TSP) as a starting problem to plan trips. Shang *et al.* propose and investigate a problem called *User Oriented Trajectory Search* (UOTS) for trip recommendation [9]. Given a trajectory data set, the query input contains a set of intended places given by the tourist and a set of textual attributes describing the tourist preference. If a trajectory is connecting/close to the specified query locations, and the textual attributes of the trajectory are similar to the tourist preference, it will be recommended to the tourist for reference.

An interesting approach to the trip recommendation problem is the one proposed by Vansteenwegen *et al.*, where authors define the Tourist Trip Design Problems (TTDP) [10, 13]. The orienteering problem, which originates in the operational research literature, is used as a starting point for modeling the TTDP. The problem involves a set of possible locations having a score and the objective is to maximize the total score of the visited locations, while keeping the total time (or distance) below the available time budget. The score of a location represents the interest of a tourist in that location. Scores are calculated using the vector space model and the TTDP is solved using a guided local search meta-heuristic. Authors compare their technique versus a competitor. Both algorithms are applied to a real data set from the city of Ghent. Results show that the approach turns out to be faster and produces solutions of better quality. Lately, they propose a tourist expert system, called the "City Trip Planner" [12], that allows planning routes for five cities in Belgium.

The orienteering problem is also employed in [2]. Here, De Choudhury *et al.* construct intra-city travel itineraries automatically by tapping a latent source reflecting geo-temporal traces left by millions of tourists. To do so, as in our case they firstly extract photo streams of individual users from Flickr. In the second step, they aggregate all user photo streams into a PoI graph. Itineraries are then automatically constructed from the graph based on the popularity of the PoIs and subject to the user's time and destination constraints. The problem is modeled as an orienteering problem and they propose a variation of a recursive greedy algorithm to solve it. Some important limitations affect the paper: i) the proposed orienteering problem does not model user preferences, ii) it also does not model co-visitation of different PoIs, iii) the greedy algorithm solving the orienteering problem explicitly needs a source PoI, a destination PoI, the total number of PoIs to be visited in the trip and a possible set of PoI to not be visited. However, this information could not be implicitly available.

Lu *et al.* [6], propose a novel data mining-based approach, namely "Trip-Mine", to efficiently find the optimal trip which satisfies the user's travel time constraint based on the user's location. Kurashima *et al.* [5] propose a travel route recommendation method that makes use of the photographers' histories as held by Flickr. Recommendations are performed by means of a photographer behavior model, which estimates the probability of a photographer visiting a landmark. Authors demonstrate the effectiveness of the proposed method using a real-life dataset in terms of the prediction accuracy of the travel behavior.

Similar to the objective of the papers cited above, our work intend to merge touristic data analysis and information synthesis. To overcome some of the limitations listed above, we propose to solve the trip planning problem in a completely unsupervised way. We exploit PoIs and their categories from Wikipedia, and the tourist traces from photos in Flickr to automatically build a dataset. This knowledge base contains very rich information about the touristic city and about the way in which tourists visited its attractions. Finally, we model the planning of a trip as an instance of the Generalized Maximum Coverage problem that composes the trip from latent tourist-generated behaviors. As a consequence, we naturally model co-visitation of different PoIs as we directly compose traces performed by real tourists in a city. Furthermore, user preferences are mapped to the categorization of PoIs automatically extracted from Wikipedia. Our method is thus able to devise when a given PoI is appealing for a particular user by mapping both the user and the PoI on a fixed categorization.

3. PROBLEM DEFINITION

Let $\mathcal{P} = \{p_1, \dots, p_N\}$ be the set of PoIs in our touristic city. Each PoI p is univocally identified by its geographic coordinates, a name, a radius specifying its spatial extent, and a *relevance vector*, $\vec{v_p} \in [0,1]^{|C|}$, measuring the normalized relevance of p w.r.t a set of categories C. Without loss of generality, we assume that the set C is predetermined and fixed and that the relevance of every PoI for each category is known. Symmetrically, let u be a user from the set $\mathcal{U}$, and $\vec{v_u} \in [0,1]^{|C|}$ the *preference vector* stating the normalized interest of u for the categories in C. The preference vector can be explicitly given by the user, or implicitly learned[1].

DEFINITION 1 (USER-POI INTEREST). *Given a PoI p, its relevance vector $\vec{v_p}$, a user u, and the associated preference vector $\vec{v_u}$, we define the* User-PoI Interest *function as a the following function $\Gamma(p,u) : \mathcal{P} \times \mathcal{U} \to [0,1]$:*

$$\Gamma(p,u) = \alpha \cdot sim(\vec{v_p}, \vec{v_u}) + (1-\alpha) \cdot pop(p)$$

where $sim(\vec{v_p}, \vec{v_u}) = \frac{\vec{v_p} \cdot \vec{v_u}}{||\vec{v_p}||\,||\vec{v_u}||}$ is the cosine similarity between the user preference and the PoI relevance vectors, $pop(p)$ is a function measuring the popularity of p, and $\alpha \in [0,1]$ is a parameter controlling how much user preference and popularity of PoIs have to be taken into account.

[1] Without loss of generality, we assume to know the preference vectors of all the users in advance. A uniform distribution can be used for $\vec{v_u}$ in the case the profile of u is (initially) unavailable.

Definition 2 (PoI History). *Giver a user u and the PoIs $\mathcal{P}$, the PoI history H_u of u is the temporally ordered sequence of points of interest visited by u. Each PoI p of H_u is annotated with the two timestamps indicating the start time and the end time of the visit:*

$$H_u =< (p_1, [t_{11}, t_{21}]), \dots (p_m, [t_{1m}, t_{2m}]) >$$

Note that having the start time and the end time we have an implicit representation of the time the user u has spent for the visit of p.

Definition 3 (Trajectory). *Given a PoI History H_u and a time threshold τ, we define a trajectory T_u any subsequence of H_u*

$$< (p_k, [t_{1k}, t_{2k}]), \dots, (p_{k+i}, [t_{1(k+i)}, t_{2(k+i)}]) >$$

such that:

$$\begin{array}{ll} i \geq 1 & \\ t_{1k} - t_{2(k-1)} > \tau, & if\ k > 1 \\ t_{1(k+i+1)} - t_{2(k+i)} > \tau, & if\ (k+i) < m \\ t_{1(k+j)} - t_{2(k+j-1)} \leq \tau, & \forall j\ s.t.\ 1 \geq j \leq i. \end{array}$$

Trajectories are thus sequences of PoIs visited consecutively. They are obtained by cutting the user PoI history where the time interval between the visit to two subsequent PoIs is greater than a given threshold.

By applying the same temporal splitting criterium to all the Poi histories of users $\mathcal{U}$ we obtain the set $\mathcal{S} = \{s_1, \dots, s_M\}$ of relevant trajectories. Note that $\mathcal{S}$ results from a set-union operation that disregards timestamps annotating every tourist visit to a PoI. The temporal information can be however exploited to compute estimates of the average time required for visiting a given PoI, and for every PoI-to-PoI movement. Thus, let $\rho(p) : \mathcal{P} \rightarrow \mathbb{R}$ be an estimate of the time needed to visit p, and $\tau(p_i, p_j) : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}$ an estimate of the time needed to a user to move from p_i to p_j. Finally, let $\vec{z} = (z_1, \dots, z_M)$ be the total traveling time associated with the M trajectories in $\mathcal{S}$, obtained by exploiting $\tau(\cdot, \cdot)$.

We are now ready to formulate the TripCover problem, i.e., the problem of generating an optimal personalized itinerary given tourist's preferences and her budget in term of available time to spend in the city. Without loss of generality, we assume that the User-PoI Interest for all the items in $\mathcal{S}$ is strictly positive. More formally, $\forall s \in \mathcal{S}$, $\sum_{p \in s} \Gamma(p, u) > 0$. In fact, if this would not hold for some trajectory, these trajectories could be removed from $\mathcal{S}$ without any consequence for the functionality of the system.

TripCover(B): Given a tourist u, PoIs $\mathcal{P}$, a time budget B, trajectories $\mathcal{S}$, User-PoI Interest function Γ, cost function $\rho(p)$ and vector $\vec{z}$. Find a subset of $\mathcal{S}$ that:

$$\begin{array}{ll} \textit{maximize} & \displaystyle\sum_{i=1}^{|\mathcal{S}|} \sum_{j=1}^{|\mathcal{P}|} \Gamma(p_j, u)\, y_{ij} \qquad (1) \\ \textit{such that} & \displaystyle\sum_{i=1}^{|\mathcal{S}|} \sum_{j=1}^{|\mathcal{P}|} \rho(p_j)\, y_{ij} + \sum_{i=1}^{|\mathcal{S}|} z_i\, x_i \leq B \\ & \displaystyle\sum_{i=1}^{|\mathcal{S}|} y_{ij} \leq 1, \quad \forall j \in \{1, \dots, |\mathcal{P}|\} \\ & \displaystyle\sum_{i=1}^{|\mathcal{S}|} x_i \geq \sum_{i=1}^{|\mathcal{S}|} y_{ij}, \quad \forall j \in \{1, \dots, |\mathcal{P}|\} \end{array}$$

where

$$y_{ij} = \begin{cases} 1 & \text{if PoI } j \text{ in trajectory } i \text{ is selected;} \\ 0 & \text{otherwise.} \end{cases}$$

$$x_i = \begin{cases} 1 & \text{if trajectory } i \text{ is selected;} \\ 0 & \text{otherwise.} \end{cases}$$

The TripCover(B) problem as formulated in (1) is an instance of the *Generalized Maximum Coverage* (GMC) problem that is proven to be NP-hard [1]. In particular, given a tourist u, TripCover(B) can be captured by the GMC formulation by: i) the bins in GMC represent the trajectories in $\mathcal{S}$; ii) the profit $\Gamma(p, u)$ and the cost $\rho(p)$ depend only on p and u and not from the bins. The TripCover(B) problem is thus NP-hard. An efficient greedy approximation algorithm for the GMC problem is known that achieves an approximation ratio of $e/(e-1) + \epsilon$, $\forall \epsilon > 0$ [1]. We used this approximation algorithm (whose source was kindly provided us by the authors) after slightly modifying it to take into account TripCover(B) specific constraints.

Scheduling the Trip on a Multi-Day Plan. TripBuilder needs to schedule the solution of the specific instance of the TripCover problem on the (multi-)day plan of the tourist. Recall that the solution of TripCover is a set of trajectories maximizing tourist profit. These trajectories have then to be scheduled on the tourist's agenda. For a lack of space, we do not address the scheduling issue in this paper. A possible solution could be to find the shortest path crossing all the starting and ending PoIs of the trajectories in the solution by means of a TSP algorithm. The TSP instance to be solved is in our case very simple since it has to devise short connections among the suggested trajectories and not among all the PoIs. The resulting solution consist in a unique path crossing all the relevant PoIs and joining all the selected trajectories that could be easily scheduled on the tourist's agenda.

4. EXPERIMENTAL EVALUATION

In this section we discuss the process used to build our touristic knowledge base from user-generated content. We also introduce the metrics used to assess effectiveness and we provide a detailed assessment of TripBuilder performance vs. meaningful baselines.

Building the Knowledge base.

In order to assess TripBuilder we consider three case studies, each with its particular characteristics. In particular, we generate - in a complete unsupervised process - a knowledge base covering three Italian cities which are important from a touristic point of view and thus guarantee variety and diversity: Pisa, Florence and Rome. The rationale of the choice is to propose a complete evaluation of our techniques by varying the size of the cities and the richness of public user-generated content available for download[2].

PoIs. The first step is identifying the set of PoIs in the target geographical region. Given the bounding box BB_{city} containing the city of interest, we download all the geo-referenced Wikipedia pages falling within this region. We assume each geo-referenced Wikipedia named entity, whose geographical coordinates falls into BB_{city}, to be a fine-grained

[2] Link to the dataset: `https://github.com/igobrilhante/TripBuilder`.

Point of Interest. For each PoI, we retrieve its descriptive label, its geographic coordinates as reported in the Wikipedia page, and the set of categories the PoI belongs to. Categories are reported at the bottom of the Wikipedia page, and are used to link articles under a common topic. They form a hierarchy, although sub-categories may be a member of more than one category. By considering the set C of categories associated with all the PoIs, we generate the normalized relevance vector of each PoI. We then perform a density-based clustering to group in a single PoI touristic entities which are very close one to each other. Clustering very close PoIs is important since a tourist in a given place can enjoy all the attractions in the surroundings even if she do not take photos to all of them. Moreover, it aims at reducing the sparsity that might affect trajectory data. To cluster the PoIs we use DBScan [3] by setting 1 as the minimum number of points and 200 meters as ϵ. Finally, we obtain the relevance vector for the clustered PoIs by considering the occurrences of each category in the members of the clusters and by normalizing the resulting vector. At the end of this first step we have the set $\mathcal{P} = \{p_1, \ldots, p_N\}$ of PoIs and the relevance vector $\vec{v_p} \in [0,1]^{|C|}$ for each of these PoIs in a fully automatic way by exploiting Wikipedia as an external source of knowledge.

Users and PoI histories. As second step we need a method for collecting users $\mathcal{U}$ and the long-term itineraries crossing the discovered PoIs. We query Flickr to retrieve the metadata (user id, timestamp, tags, geographic coordinates, etc.) of the photos taken in the given area BB_{city}. The assumption we are making is that photo albums made by Flickr users implicitly represent touristic itineraries within the city. To strengthen the accuracy of our method, we retrieve only the photos having the highest georeferenced precision. This process thus collects a large set of geo-tagged photo albums taken by different users within BB_{city}. We preliminary discard photo albums containing only one photo. Then, we spatially match the remaining photos against the set of PoIs previously collected. We associate a photo to a PoI when *the photo was taken within a circle having the PoI as its center and* $r = 100$ *meters as radius.* Note that in order to deal with clustered PoIs, we consider the distance of the photo from all constituent members: in the case the photo falls within the circular region of at least one of the members, it is assigned to the clustered PoI. Moreover, since several photos by the same user are usually taken close to the same PoI, we consider the timestamps associated with the first and last of these photos as the starting and ending time of the user visit to the PoI. Finally, the popularity of each PoI is computed as the number of distinct users that take at least one photo in its circular region. The above process allows us to generate the set of users $\mathcal{U}$, their PoI history, and estimates for the popularity and visiting time of each PoI. Finally, the preference vector for each user is built by summing up and normalizing the relevance vectors of all the PoIs occurring in her PoI history.

Trajectories. In order to build the set $\mathcal{S}$ of trajectories we split users' PoI histories as detailed in Definition 2. To choose the splitting threshold τ we try to understand users' macroscopic behavior by carefully analyzing the inter-arrival time of each pair of consecutive photos taken in different PoIs. Therefore, for each city we compute the distribution of probability of the inter-arrival time $P(x \leq \tau)$ of pairs of consecutive photos. Then we devise the time threshold τ such that $P(x \leq \tau) = 0.9$. Results show that while for Rome and Florence the resulting threshold is about 5 and 6 hours respectively, for the smallest city of Pisa it decreases to about 3 hours.

Traveling time computation. An important aspect of TripBuilder is that we recommend complete itineraries fitting the available time budget and not just the set of PoIs to be visited. The trip building step should therefore consider not only PoI visiting time but also the time $\tau(p_i, p_j)$ needed to move between consecutive PoIs in the itinerary. Since measuring intra-PoI moving time from the photo albums resulted to be inaccurate for not popular PoIs, we resort to an external service. Given two PoIs in a city, we compute function $\tau(\cdot,\cdot)$ by querying for the Google Maps' walking distance. Naturally, this is an approximation since several variations could happen: tourists may use a car, or use public transportation, or take a taxi. However, our method is parametric to these aspects, and the system can be easily adapted to consider the different choices. Moreover, most PoIs in our touristic cities are actually at walking distances.

Table 1 shows the main characteristics of the three datasets. The second column reports the number of PoIs for each of the three cities. Note that these numbers refer to the result of the clustering phase, while the number of entities extracted from Wikipedia are 124, 1,022, and 671 for Pisa, Florence and Rome, respectively. Furthermore, columns "# Users" and "# Photos" report the number of distinct users and photos retrieved from Flickr. Finally, column "# Traj." reports the total number of trajectories extracted from the dataset.

City	# PoIs	# Users	# Photos	# Traj.
Pisa	112	1825	18,170	3,430
Florence	891	7049	102,888	16,522
Rome	490	13772	234,616	35,522

Table 1: Statistics regarding the three datasets.

We assess the effectiveness of TripBuilder by comparing its performance with those obtained by two baseline methods on common evaluation metrics that consider the actual behavior of users as mined from Flickr. In particular, we conduct our experiments on the three cities by splitting the three datasets in *training* and *test* sets, and by varying the parameter α affecting the contributions of PoIs/user-similarity and PoI-popularity to user profit. For every city, we consider the 100 users with the longest PoI histories as test set. Since users in the test set are the golden standard used to compute effectiveness figures, we choose the users having the longest PoI histories to be able to vary in a significant range the time budgets (e.g., we can not evaluate a personalized 4-days itinerary in Rome with test users that actually visited only a pair of popular PoIs). The average number of PoIs visited by our 100 test users is 12.2, 53.9, 51.2 for Pisa, Florence, and Rome, respectively. Note that in the whole datasets the averages drop respectively to 2.7, 4.9, and 4.6. The preference vector of every one of the 100 test users in each city, along with a time budget varying in the range 1, 2, and 4 days (1/2, 1 days in the case of the small city of Pisa that can be visited even in less than one day)[3], are given in input to TripBuilder and the baseline algorithms.

Baselines algorithms.

We compare the performance of TripBuilder against the baselines detailed below.

Trajectory Popularity (Tpop). This baseline builds the trip by taking into account the normalized popularity of the trajectories in $\mathcal{S}$ computed as the sum of the popularity of the constituent PoIs divided by the length of the trajectory. It works by adding to the visiting plan once at a time the most popular trajectories until the time budget is reached.

Trajectory Personalized Profit (Tppro). Given the preference vector of a tourist, this baseline sorts the trajectories in $\mathcal{S}$ by decreasing normalized user/PoI similarity. Such trajectory score is computed as the sum of user/PoI similarities of all the PoIs in the trajectory divided by trajectory length. The baseline algorithm builds the personalized itinerary by adding once at a time the trajectories having the highest profit for the specific tourist until the total time budget is reached.

Performance metrics.

The metrics used to evaluate and characterize the itineraries are defined below.

Recall (on PoIs and Categories). This is the popular recall metrics that in the Information Retrieval domain measures the fraction of the documents that are relevant to the query that are successfully retrieved. In our case it is computed for a user and a suggested itinerary as the fraction of PoIs (or Categories) in the user PoI history which occurs in the suggested itinerary.

Popularity Score. A popularity score S^{pop} is computed for an itinerary T by summing the popularity of the PoIs covered by T. More formally,

$$S^{pop}(T) = \sum_{p_i \in T} pop(p_i)$$

where $pop(p)$ is the number of distinct users who visited p.

Personal Profit Score. Given an itinerary T and a user u, a profit score (S_u^{pro}) for T can be computed as

$$S_u^{pro}(T) = \sum_{p_i \in T} sim(\vec{v_p}, \vec{v_u})$$

where $sim(\vec{v_p}, \vec{v_u})$ is the user/PoI similarity function over PoI relevance user preference vectors given in Definition 1.

Visiting Time Score. This score assumes that the average per-PoI visit time of an itinerary is related with its interestingness. The higher the average per-PoI visit time computed as the sum of the visiting times for the constituent PoIs, the higher the interestingness. Given an itinerary T, its Visiting Time Score (S^{vt}) is

$$S^{vt}(T) = \sum_{i=1}^{k} \rho(p_i)$$

where $\rho(p)$ is the average visiting time (in seconds) for PoI p. Note that this metrics is particularly meaningful for budgeted itineraries as the ones we deal with. In fact at parity of total time budget an itinerary with higher Visiting Time Score should be preferred over one having a lower score since the former in principle involves more time to enjoy interesting attractions and less time to move from one PoI to the next one.

Experimental Results.

The results of the experiments conducted are reported in Table 2. It is worth recalling that our approach aims at maximizing the user's total profit/interest over the PoIs fitting her budget. In terms of S_u^{pro}, our solution improves the baselines up to 91% in Pisa (with an absolute improvement ΔS_u^{pro} of 0.272), 173% in Florence ($\Delta S_u^{pro} = 0.249$) and 130% in Rome ($\Delta S_u^{pro} = 0.403$). In addition, it builds trips that increase S^{vt} up to 25 minutes in Pisa, about 4 hours in Florence, and approximately 11 hours in Rome. Therefore our algorithm suggests itineraries that better match user preferences and involve lower intra-PoI movement time than the baselines.

By observing the column S_u^{pro} on Table 2, we can conclude that TripBuilder constantly outperforms the baselines and presents a behavior which is sensitive to α. It is worth highlighting two situations: i) when $\alpha = 0$, TripBuilder is comparable to Tpop (both considering only popularity), and we can see that our algorithm obtains higher S_u^{pro}; ii) when $\alpha = 1$, TripBuilder considers only user's interest, similarly to Tppro, but still it achieves higher S_u^{pro} values. Thus, we may conclude that α plays an important role in TripBuilder to balance the contribution of user's profit/interest and PoI's popularity.

Another important metric is the visit time reported in column S^{vt}. The higher the visit time, the more relevant the recommended trip is to the user, since it is more advantageous to the user to spend time visiting the PoIs than moving among them. As our GMC formulation takes this factor into account (as a cost), it tends to exploit in the solution trajectories that visit close PoIs and maximize user profit. Consequently, TripBuilder is able to globally build trips maximizing S^{vt}. We can see from the results in the table that TripBuilder uses more appropriately the time budget. The difference in S^{vt} becomes higher when bigger cities and larger budgets are considered. In the case of Pisa, the three algorithms have quite similar Visit Time Scores, with slight gains for TripBuilder. In the case of the larger cities of Florence and Rome, TripBuilder remarkably outperforms the baselines. This could happen because PoIs in small cities are close and concentrated in a small region, while in bigger cities larger intra-PoI traveling time can impact the S^{vt} metric.

In terms of PoIs and categories recall (Recall-P and Recall-C in Table 2), all algorithms get at least 75% of the relevant PoIs and 96% of the categories for Pisa. As shown in Table 2, TripBuilder cannot reach the 99% of category recall of Tpop, but the 98% is still a signal that TripBuilder chooses PoI of relevant categories for the users. Looking at PoIs recall, on the other hand, TripBuilder gets better results than the baselines: 87% compared to 83% of Tpop and 79% of Tppro for the one-day time budget. When we compare the results for Florence and Rome, we observe that TripBuilder outperforms PoIs recall results as well as cat-

[3] We assume the normal daily activity of a tourist in a city to be of twelve hours. Our solution is, however, completely agnostic w.r.t. the daily agenda and works as well with tourist-provided agenda defining different time slots.

Pisa						
	Days	Recall-P	Recall-C	S_u^{pro}	S^{vt}	S^{pop}
Tpop	½	0.480	0.755	0.298	14443	0.548
	1	0.833	0.990	0.609	28984	0.874
Tppro	½	0.560	0.803	0.391	14535	0.576
	1	0.797	0.962	0.618	28272	0.821
TB, (0)	½	0.712	0.910	0.391	16086	0.798
	1	0.822	0.988	0.601	28968	0.876
TB, (0.5)	½	0.725	0.904	0.565	16027	0.673
	1	0.863	0.984	0.709	29452	0.890
TB, (1)	½	0.721	0.898	0.570	15931	0.648
	1	0.871	0.984	0.715	29510	0.880

Florence						
	Days	Recall-P	Recall-C	S_u^{pro}	S^{vt}	S^{pop}
Tpop	1	0.303	0.791	0.085	30456	0.437
	2	0.380	0.843	0.144	53523	0.505
	4	0.545	0.931	0.299	104839	0.654
Tppro	1	0.276	0.751	0.102	29434	0.404
	2	0.371	0.840	0.184	53589	0.489
	4	0.545	0.926	0.322	104805	0.643
TB, (0)	1	0.529	0.886	0.183	33450	0.656
	2	0.662	0.924	0.300	64021	0.761
	4	0.767	0.965	0.449	118944	0.842
TB, (0.5)	1	0.444	0.836	0.269	30932	0.465
	2	0.617	0.917	0.393	59683	0.668
	4	0.764	0.960	0.519	120389	0.814
TB, (1)	1	0.441	0.832	0.269	30844	0.460
	2	0.613	0.916	0.393	59607	0.664
	4	0.762	0.959	0.519	120247	0.812

Rome						
	Days	Recall-P	Recall-C	S_u^{pro}	S^{vt}	S^{pop}
Tpop	1	0.386	0.783	0.161	19890	0.523
	2	0.534	0.896	0.310	39277	0.665
	4	0.620	0.934	0.431	76025	0.727
Tppro	1	0.316	0.783	0.174	17286	0.397
	2	0.500	0.873	0.353	36135	0.596
	4	0.618	0.932	0.486	69309	0.695
TB, (0)	1	0.652	0.886	0.337	33609	0.764
	2	0.838	0.959	0.579	61644	0.891
	4	0.910	0.991	0.729	109276	0.944
TB, (0.5)	1	0.566	0.750	0.512	31941	0.529
	2	0.831	0.945	0.712	61994	0.802
	4	0.936	0.989	0.818	108767	0.943
TB, (1)	1	0.561	0.740	0.512	31854	0.520
	2	0.825	0.943	0.713	61874	0.795
	4	0.935	0.989	0.820	107773	0.939

Table 2: Performance of TripBuilder (TB) by varying the parameter α and the baselines according to various metrics.

egories recall results in all the experiments. This happens mostly thanks to the capability of TripBuilder of building trips with an higher visit time within the time budgets, consequently more PoIs are likely to be visited. Finally, we can see that the α parameter allows to fit the expectations of the user. For small values of α we have higher recall values because trajectories crossing popular PoIs are preferred. When α is increased, recall figures decreases (although stay always significantly over the baselines) because unexpected trajectories fitting the user interests are suggested that may constitute serendipitous recommendations.

5. CONCLUSIONS AND FUTURE WORK

In this paper we introduced TripBuilder, a unsupervised framework for the recommendation of personalized touristic itineraries that model the planning of a trip as a instance of the Generalized Maximum Coverage problem. It works by composing the itinerary that maximizes a measure of user interest over the PoIs while globally respecting the user time budget. We evaluate TripBuilder on datasets collected for three cities differentiated by size and touristic interest. Results show that TripBuilder outperforms two strong baselines for all the metrics adopted in the assessment. Future work includes a deeper investigation of how to schedule the TripBuilder solution on the tourist agenda.

Acknowledgments. We acknowledge Liran Katzir and all authors of [1] for providing us their GMC source code. This work was partially supported by EU FP7 Marie Curie project SEEK (no. 295179), CIP-PSP project E-CLOUD (no. 325091), PRIN 2011 project ARS TECNOMEDIA, CNPQ Scholarship (no. 306806/2012-6), CNPQ Casadinho/ PROCAD Project (no. 552578/2011-8), and CNPQ-CNR Bilateral Project (no. 490459/2011-0).

References

[1] R. Cohen and L. Katzir. The generalized maximum coverage problem. *Information Processing Letters*, 108(1):15–22, 2008.

[2] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In *Proc. ACM HT*, 2010.

[3] M. Ester, H.P. Kriegel, J. S, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[4] J.-M. Godart. Combinatorial optimisation based decision support system for trip planning. In *Information and Communication Technologies in Tourism*, pages 318–327. Springer, 1999.

[5] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura. Travel route recommendation using geotags in photo sharing sites. In *Proc. ACM CIKM*, 2010.

[6] E. Lu, C. Lin, and V. Tseng. Trip-mine: An efficient trip planning approach with travel time constraints. In *Proc. IEEE MDM*, 2011.

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

[8] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *Proc. ACM KDD*, 2009.

[9] S. Shang, R. Ding, B. Yuan, K. Xie, K. Zheng, and P. Kalnis. User oriented trajectory search for trip recommendation. In *Proc. ACM EDBT*, 2012.

[10] W. Souffriau, P. Vansteenwegen, J. Vertommen, G. Berghe, and D. Van Oudheusden. A personalized tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence*, 22(10):964–985, 2008.

[11] P. Vansteenwegen and W. Souffriau. Trip planning functionalities: state of the art and future. *Information Technology & Tourism*, 12(4):305–315, 2010.

[12] P. Vansteenwegen, W. Souffriau, G. Berghe, and D. Oudheusden. The city trip planner: an expert system for tourists. *Expert Systems with Applications*, 38(6):6540–6546, 2011.

[13] P. Vansteenwegen and D. Van Oudheusden. The mobile tourist guide: an or opportunity. *OR Insight*, 20(3):21–27, 2007.

[14] H. Yoon, Y. Zheng, X. Xie, and W. Woo. Smart itinerary recommendation based on user-generated gps trajectories. *Proc. IEEE UIC*, LNCS, 2010.

[15] H. Yoon, Y. Zheng, X. Xie, and W. Woo. Social itinerary recommendation from user-generated digital trails. *Personal and Ubiquitous Computing*, 16(5):469–484, 2012.