Personalized Travel Suggestions for Tourism Websites

António Coelho
INESC Porto/Departamento de Engenharia Informática
Faculdade de Engenharia, Universidade do Porto
Porto, Portugal
acoelho@fe.up.pt

André Rodrigues
INESC Porto
Porto, Portugal
andre.c.rodrigues@inescporto.pt

***Abstract*—The evolution of tourism websites is converging to a set of features and best practices that are becoming standard, in such a way that developing a template and later customizing it to a given tourist region is becoming feasible. A very important feature in this kind of site is showing the touristic suggestions of what the tourist can find in the destination, optimally personalized for him/her. One problem in deploying such functionality however is the lack of user experience data suited to perform data mining when a new site is launched. This paper proposes a solution, customizable to any touristic region, that harnesses the information available in Flickr, crossing it with a Point of Interest (POI) database and using Google Prediction API (Application Programming Interface) to generate personalized travel suggestions, based on the geographical itinerary the user defined with a trip planner tool.**

Data mining; Geographical Information Systems (GIS); machine learning; cloud computing; recommendation systems

I. INTRODUCTION

One of the main goals of a travel website is helping the potential tourist to discover places to visit when planning a vacation to a touristic region, be it a city, a region or a country. Upon the first interaction with the website, there's no information regarding the user on which suggestions can be based but, as the browsing continues, more and more data becomes available, and the creation of personalized recommendations gradually becomes feasible.

This paper focuses on a particular tool that's common to many travel websites: the *trip planner*. As the user browses through several POI, like a monument, a viewpoint or a restaurant, the option "Add to My Trip" is available on each item, allowing the building of an itinerary using the *shopping cart* [10] design pattern. Further options allow reordering the visit sequence and removing POI.

This paper discusses the usage of the built itinerary in producing customized recommendations, based on the proximity, the explicit profile of the user and his/her inferred profile, using the Douro Wine Region in Portugal as a case study.

Next section presents related work, followed by a description of the goals and methodology of the solution presented. Section IV presents the suggestion engine on an incremental basis and section V describes how to populate the database with rankings associated to POI without having registers of the user experience. Section VI explains how to get recommendations using the Google Prediction API and section VII shows how to consider the driving distance. Finally, section VIII describes the conclusions and future work.

II. RELATED WORK

One of the best known example of a recommendation engine is Amazon's *Customers Who Bought This Item Also Bought…* feature, which assumes the user may be interested in items related to the one currently being explored, based on the analysis of other customers buying patterns [6]. Although it doesn't require the user to explicitly rate a number of items to build a profile, it's based on only one item. Hence, if applied in the context of a trip planner tool, it wouldn't consider the set of selected POI as a whole, and could only be used for the generation of a suggestion for each point of interest selected by the user. Additionally, it completely ignores useful available geographical information.

Maybe the simplest implementation of a recommendation engine is the *slope one* family of algorithms [5], but again its applicability on a tourism website is very limited due to its ratings based approach: if a user visits the website while preparing the first trip to a certain region, there is no previous records to rate it and location based information is also ignored.

A geographical based approach is taken by Clements, Serdyukov, de Vries and Reinders [1]. If a user uploaded to Flickr a set of geotagged photos of a first city, suggestions of places to visit in a second city are generated based on the analysis of the geotags of photos from other users that visited both cities and who showed similar tastes regarding the places visited on the first city. Only the 10 most visited cities worldwide were considered, which is very limiting; additionally, the requirement that there must be geotagged photos of a city that have been previously uploaded severely restricts the population it can cater to.

Another geographical approach was taken by Kunczier, Michlmayr, Pospischil and Anegg [7], but on a mobile device. The *Lol@* system lists points of interest on the proximity of the user. There are plans to choose POI considering the profile of the user, that must be defined explicitly nonetheless.

Google also ends up being a personalized recommendation engine as it selects some search results to appear first, based both on how each result ranks on an internal measure (the PageRank) and on how relevant it may be to the inferred profile, based on past behaviour [3].

978-1-4577-1676-8/11/$26.00 ©2011 IEEE

Pertinent is the issue of user profiling. Mencar et al. [8] propose a strategy based on fuzzy logic and continued adaptation to the history of views. They begin by proposing the compatibility between the structure and profile of the item, so that it is possible to determine which items match a given profile, within a tolerance value, as the user navigates through items. Besides, those who were seen further in the past begin to have less weight in determining the profile and those who were seen more recently become more relevant. Alternatively, the construction of the profile can combine both a plastic period, which is more sensitive to updates, and a stable period, less sensitive. One advantage of this approach is that the construction of the profile is implicit, based only on the user behaviour but, on the other hand, requires that all items be properly classified, which makes it unfeasible sometimes. The survey compiled by Godoy and Amandi [2] shows most lines of research divide this process in four main dimensions: acquisition, learning, adaptation and evaluation; it also focuses the representation aspect, noting that it ranges from a simple set of keywords to complex structures output from learning algorithms; an overlap with the fields of categorization and learning is also recognized in many implementations; promising development seems to involve semantic characterization in addition to temporal and contextual aspects consideration.

III. Goals and Methodology

The main goal of this work is to provide suggestions of interesting places to a trip planner tool, shown in Fig. 1, which considers the POI chosen by the user in her planned itinerary, both regarding its category and location within the region.

Another possible use is to give suggestions to a user browsing a POI details page, where the previous points of interest added to "My Trip" together with the item currently being viewed can be considered in the suggestion process.

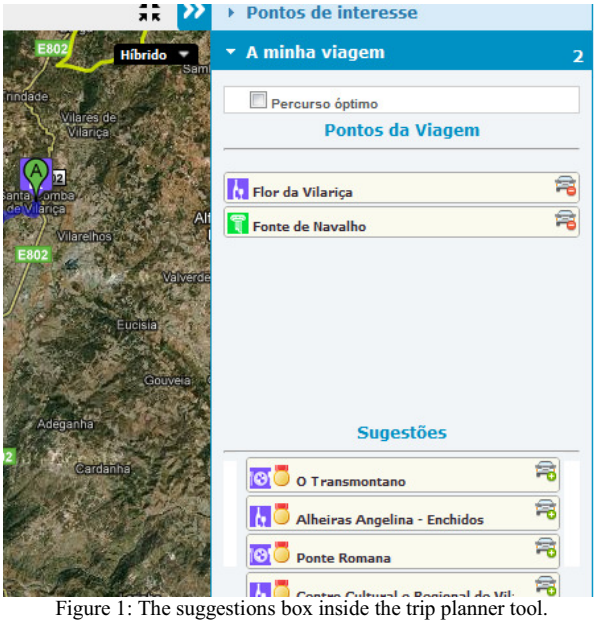

Figure 1: The suggestions box inside the trip planner tool.

An iterative approach is being used as the method to generate personalized travel suggestions, where subsequent iterations build on the previous ones. In particular, component values associated with each analyzed characteristic and with a given weight are combined in the end to sort the possible results, and the top ones are shown to the user. This is inspired by [9], where different solutions are linearly combined in order to determine the best suggestions to be given to the user.

For example, after the third development cycle, the score of POI_i is the sum of the partial scores given in the components associated with each iteration, affected by a coefficient, which is its weight, as shown in (1).

$$s_i = w_1 c_{1i} + w_2 c_{2i} + w_3 c_{3i} \quad (1)$$

Where s_i is the score of POI_i; c_{1i}, c_{2i} and c_{3i} are the partial scores POI_i obtained in each component and the coefficients w_1, w_2 and w_3 are each component's weights. Thus, a generalisation can be thought of as in (2).

$$s_i = \sum_{j=1}^{n} w_j c_{ji} \quad (2)$$

Where c_{ji} is the partial score POI i got at characteristic j.

IV. The Suggestion Engine

To bootstrap the iterative development process, one can use the naive, yet simple method of selecting random points. This enables the focusing on the structure and interoperability of the recommendation component. To follow the proposed methodology, a random number is associated with each POI (excluding services, due to their inherent lack of touristic interest), the resulting list is sorted according to this value, the points of the route of the user are removed (to avoid suggesting a POI already chosen by the user) and the highest ranking ones are selected. An output of an execution is shown in Fig. 2.

Besides its simplicity, an additional benefit is the fact that each execution provides a different set of results. If the suggestion list displayed to the user is refreshed periodically or

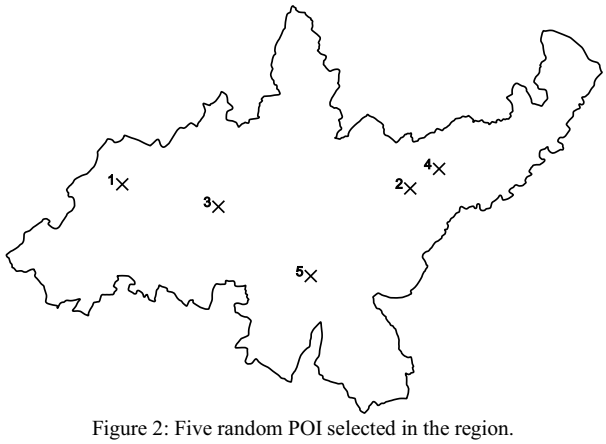

Figure 2: Five random POI selected in the region.

at each change made to the itinerary, the user is exposed to a greater number of possibilities without sacrificing GUI space or adding complexity, something that would not be possible if a fixed criterion for sorting had been used.

However, the previous method does not take into account the route defined by the user, suggesting points of interest far away from it. Thus, a first step might be to restrict the POI to those that are within a maximum distance of the route.

Relevant to this goal is the spatial operation defined by the OGC (Open Geospatial Consortium) as the method STBuffer() that generates an area involving the geometry that distances no further than a specific maximum distance.. An example is the application to a point, which generates a circle whose radius is the given margin; another example is applying it to the shoreline of a country with a margin of 200 nautical miles, yielding its EEZ (Exclusive Economic Zone).

The first step to take in order to use the method is the construction of a shape based on the sequence of POI the user wants to visit. Then the random version is adapted to only consider POIs within the area returned by the method STBuffer() applied to the constructed shape. An output of this implementation can be seen in Fig. 3, where a user selected points in Vila Real, Peso da Régua and Pinhão. It can be seen that all suggestions are in the proximity of the imaginary line connecting them.

After the considerable reduction of the search space to the POI within a maximum distance of the route, a further refinement can be made by prioritizing points of interest closer to the route, using another OCG method: STDistance(). The ordering that precedes the choice of the POI takes into account the components *distance* (expressed in a value between 0.0 and 1.0, inversely proportional to distance) and *random* (also with values between 0.0 and 1.0) with default weights 2.0 and 1.0, respectively, following the methodology mentioned in Section IV.

When defining a profile, the user can explicitly set her taste on several categories. As an example, in the case study four categories were defined (called *pillars*): Tourism, Landscape, Wine and Culture & Heritage (as shown in Fig. 4). The interface provides sliders that allow the selection of a discrete value between 0 and 3.

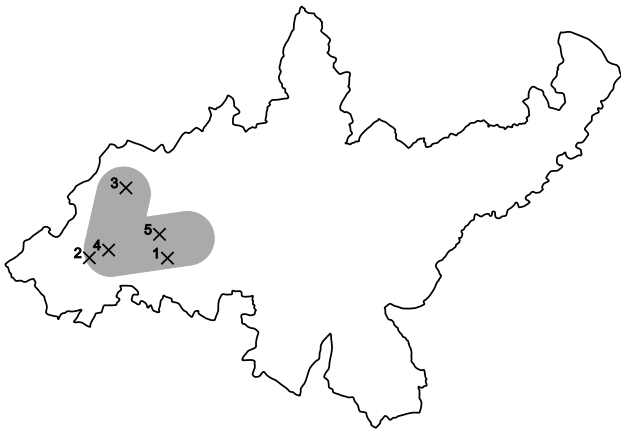

Figure 3: Five random POI, this time restricted to a given maximum distance from the itinerary.

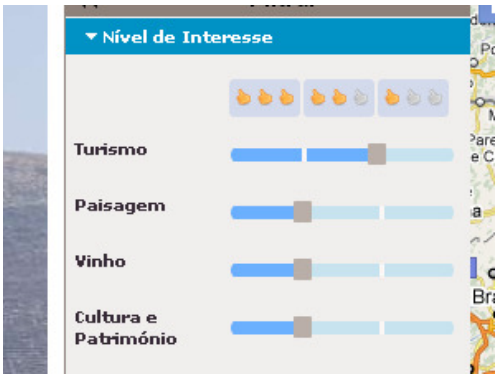

Figure 4: The explicit definition of the user profile is achieved using four sliders.

The way this preference is considered is similar to the other components: each POI gets assigned a value between 0.0 and 1.0 based on the pillar that categorizes the point of interest and the value corresponding to the user preference regarding that pillar. This value is then combined with the two other components (random and distance) with a default weight of 5.0.

V. Populating the Database Before Launch

A problem when a new tourist site becomes available online is the lack of travel planning history, making it impossible to perform any data mining. One possible way to solve this is by using information freely available on the Web as, e.g., in Flickr, which, besides providing geographic information regarding many photos, also stores the user who took them, as well as date and time.

The Flickr API allows searches based on a bouding box, and the returned photo locations can be further trimmed using the borders of the municipalities to fit the precise boundary of the region.

Yet another problem derives from *geotagging in post-processing*. Some software exists that enables adding geographic location to a photo *a posteriori*, by performing *geocoding*. But when there's very little info, e.g. just the name of a village, many pictures end up all having the centre of that village as the geotag. Though the API documentation states the geotag precision can be queried, the fact is that all pictures in the region report the same value: *maximum precision*, thus making this information useless. This problem can be mitigated by removing all photos that share a geotag more than six times, a number obtained via trial and error, and probably needs a reassessment on a different region (a urban one instead, for example).

With these data, it is possible to reconstruct each route. Fig. 5 illustrates 115 routes derived from that site.

Using a POI database, it is possible to associate each photo to its nearest point of interest. On one hand, pairings that wouldn't otherwise be obtained (like "Monument to the World War Fighters"/"Unknown Soldier Statue", having no common words) were found; but as the distance increases, the success rate decreases.

Considering sharing a word in the name as a success (the opposite doesn't work, as a photo with an automatically

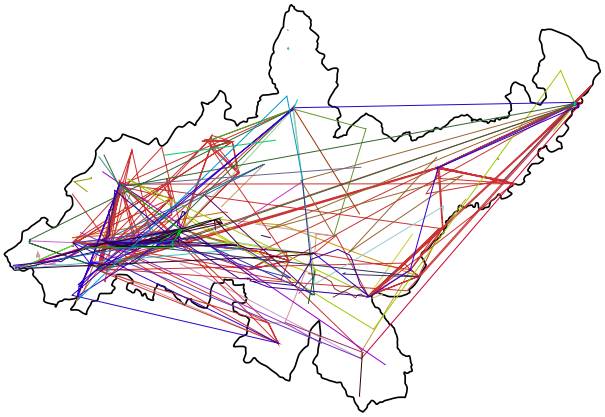

Figure 5: The reconstructed routes from the Flickr database, each in a different tone.

generated name cannot be considered a failure if it doesn't share a word with the associated POI) and plotting the accumulated successes percentage against the distance, on Fig. 6, it's observable that beyond the first 80%, the pairing method drastically loses effectiveness. Therefore, the corresponding distance, 99.4 m, can be used as a threshold to determine if a matching is valid.

This pairing allows the determination of the category of each step in a reconstructed itinerary, using the POI database.

VI. Recommendations Using Google Prediction API

Using machine learning techniques, it's now possible to try generating suggestions, starting from these categorized routes.

Google Prediction API, now in its 1.2 release, is becoming stable and mature enough that it can be used for these particular tasks. One way to train it can be to specify a POI from a route as the answer to the remaining POI of that route. The supposition is that someone who visits (p_1, p_2, p_3, p_4) recommends p_1 to those who visit (p_2, p_3, p_4)..

Unfortunately, the tests performed show that the low number of rebuilt itineraries doesn't allow recommendations

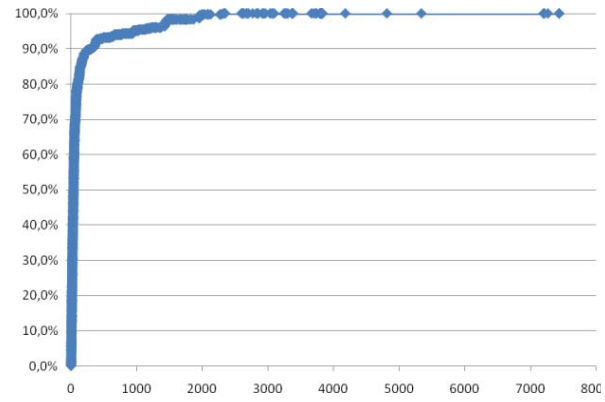

Figure 6: Graph plotting the accumulated percentage of successes against the distance (in meters).

Table 1: Report at the end of a training session in Google Prediction API.

Kind		prediction#training
Id		feupmieic/treino.csv
Self Link		https://www.googleapis.com/prediction/v1.2/training/%s
Model Info	**Model Type**	classification
	Classification Accuracy	55%
Training Status		DONE

with this degree of granularity (0% reported accuracy was obtained), neither did the ones relating subcategories (33% reported accuracy was obtained). However aiming just for the category of a POI as the answer to the proportion of each category in the remaining POI of the route yields 55% reported accuracy, as can be seen in a training output presented in Table 1 (formatted for readability), and the results can be used in the recommendation process, with a low weight.

Considering that to the itinerary (p_1, p_2, p_3, p_4), the categories (cat_7, cat_3, cat_7, cat_5) correspond, the training would be given by these data:

$$cat_7 \leftarrow (0, 0, \tfrac{1}{3}, 0, \tfrac{1}{3}, 0, \tfrac{1}{3}, 0, 0, 0)$$
$$cat_3 \leftarrow (0, 0, 0, 0, \tfrac{1}{3}, 0, \tfrac{2}{3}, 0, 0, 0)$$
$$cat_7 \leftarrow (0, 0, \tfrac{1}{3}, 0, \tfrac{1}{3}, 0, \tfrac{1}{3}, 0, 0, 0)$$
$$cat_5 \leftarrow (0, 0, \tfrac{1}{3}, 0, 0, 0, \tfrac{2}{3}, 0, 0, 0)$$

Exemplifying what was done in the case of p_2, it is recommended in the case of a visit to p_1, p_3 and p_4; to these, the categories cat_7, cat_7 and cat_5 correspond; thus cat_7 has a proportion of ⅔, cat_5 of ⅓ and the remaining ones, of zero; hence the training is performed with the sequence (0, 0, 0, 0, ⅓, 0, ⅔, 0, 0, 0) yielding cat_3, which is p_2's category).

A benefit of the answers given by the Google Prediction API is that they detail confidence to each category, as can be seen the output of the prediction for (0, 0, ⅔, ⅓, 0, 0, 0, 0, 0, 0), presented in Table 2 (formatted for readability). So, even though cat_7 is the answer, it's possible to inspect how likely the other categories were. This enables giving each POI another component, based on its category and the answer from this API.

Table 2: Report of a prediction where cat_7 is the answer.

Kind	prediction#output	
Id	feupmieic/treino.csv	
Self Link	https://www.googleapis.com/prediction/v1.2/training/%s/predict	
Output Label	cat_7	
Output Multi	**Label**	**Score**
	cat_8	7.5452 %
	cat_5	8.0378 %
	cat_1	23.9386 %
	cat_7	32.9607 %
	cat_2	8.6684 %
	cat_4	2.8882 %
	cat_6	1.2912 %
	cat_9	14.2919 %
	cat_3	0.3780 %

VII. Considering the Driving Distances

Having reached an ordered set of POI according to a classification that takes into account the weighted average of all components, the driving distances must finally be considered, as opposed to as the crow flies ones, used up until now. These are particularly important in the case study (its geography is mainly a valley with a big divide: the Douro river) as, though two points located in different banks of the river are separated by a few dozen meters, if there isn't a nearby bridge, the distance turns out to be much higher on the road.

However, while the latter are computationally very fast, the former requires a comparatively higher calculation time. Following the practice of using the infrastructure associated with Google Maps, in order to obtain consistent results, there's additionally the lag caused by the queries to Google's remote *distance calculation server*. Additionally, the use of this other API is severely restricted, allowing a maximum of 2500 requests per day in the free version [4].

Some measures can be taken to minimize these problems, including using a cache storing calculated distances, making simplifications like assuming that the distance between the origin and destination is the same as between the destination and origin (i.e., ignore one way roads) and decrease the number of POI considered to a reduced set (30, for example), but the unavailability of the service must always be considered.

If the distance calculation service is not available, or a limit has been reached, an estimation can always be given based on the geodesic distance and the driving distances stored in the database, as shown in (3).

$$\mathrm{d}'(p,q) \approx \mathrm{d}(p,q)\frac{\sum_{i=1}^{n}\mathrm{d}'(p_i,q_i)}{\sum_{i=1}^{n}\mathrm{d}(p_i,q_i)} \tag{3}$$

Where d(p, q) and d'(p, q) represent the geodesic and driving distances, respectively, between points p and q; pairs of points in cache are represented by p_i and q_i.

The way to determine the required deviation to an itinerary in order to visit a suggestion must also be considered. Fig. 7 shows an example, where the user route is the solid line and the dashed route is the minimum deviation required to visit s_1.

This value is easy to calculate using (4).

$$\mathrm{d}'(p_3,s_1)+\mathrm{d}'(s_1,p_4)-\mathrm{d}'(p_3,p_4) \tag{4}$$

And a generalization can be found in (5).

$$\min_{i=1}^{n-1}\left(\mathrm{d}'(p_i,s_j)+\mathrm{d}'(s_j,p_{i+1})-\mathrm{d}'(p_i,p_{i+1})\right) \tag{5}$$

And considering the cases where the minimum deviation takes place by visiting the suggestion before and after the itinerary, (6) can be determined.

$$\min\left(\mathrm{d}'(s_j,p_1),\min_{i=1}^{n-1}\left(\mathrm{d}'(p_i,s_j)+\mathrm{d}'(s_j,p_{i+1})-\mathrm{d}'(p_i,p_{i+1})\right),\mathrm{d}'(p_n,s_j)\right) \tag{6}$$

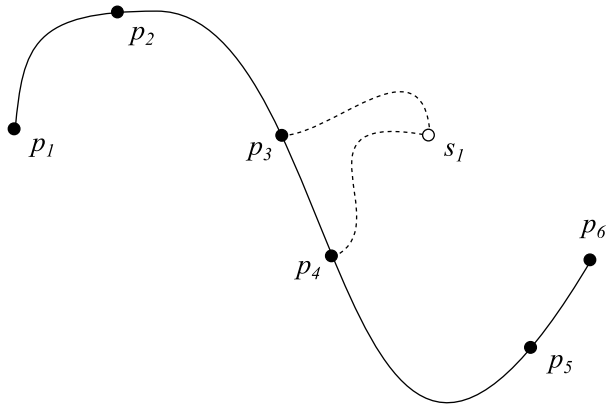

Figure 7: The deviation from an itinerary required to visit suggestion s_1.

The smallest deviation can then be used together with each POI, for the final sorting, in conjunction with the remaining components.

VIII. Conclusion and Future Work

The proposed solution achieves a satisfaction of all proposed goals, by mixing very different approaches. The usage of a cloud computing API together with selected server features and expecting extensive use of client functionalities of a browser proves to be a very powerful combination.

Though it builds on part of the work by Clements, Serdyukov, de Vries and Reinders [1], a very distinct approach is taken, not requiring the user to have previous uploaded photos in Flickr; it's also more flexible, not being restricted to the 10 most visited cities, in part due to the use of a POI database, hence innovating on this path.

Possible improvements may come from buying a larger points of interest database (e.g., of the entire country) and performing the training with the bigger dataset; this can however bring data from tourists with a very different profile (*wine tourism* in the Douro Region vs. *sun and beach tourism* in the Algarve). Similar regions (e.g., Champagne in France or Napa Valley in California, USA) may mitigate this problem, but introduce others, like possible incompatibility of the database schema used.

The dependencies on external services may benefit of a fallback upon their unavailability. Google Prediction API may be backed by Weka or Apache Mahout; the driving distance estimations could be replaced by the stack OSM (Open Street Maps) + PostgreSQL + PostGIS + pgRouting.

References

[1] M. Clements, P. Serdyukov, A.P. de Vries, M.J.T. Reinders, "Using flickr geotags to predict user travel behaviour," Proc. 33rd international ACM SIGIR conference on Research and development in information retrieval (SIGIR '10), 2010, pp. 851-852

[2] D. Godoy and A. Amandi, "User profiling in personal information agents: a survey," In The Knowledge Engineering Review, vol. 20, no. 04, 2005, pp. 329-361, doi:10.1017/S0269888906000397

[3] Google Inc., Personalized Search Basics, http://www.google.com/support/accounts/bin/answer.py?answer=54041

[4] Google Inc., Google Distance Matrix API Usage Limits, http://code.google.com/apis/maps/documentation/distancematrix/#Limits

[5] D. Lemire and A. Maclachlan, "Slope One predictors for online rating-based collaborative filtering," In SIAM Data Mining (SDM'05), Newport Beach, California, USA, April. 2005, arXiv:cs/0702144v2

[6] G. Linden, J. Jacobi and E. Benson, "Collaborative recommendations using item-to-item similarity mappings", US Patent 6,266,649 (to Amazon.com), Patent and Trademark Office, Washington, D.C., USA, 2001

[7] H. Kunczier, E. Michlmayr, G. Pospischil and H. Anegg, "LoL@ – a prototype of a network independent wireless internet service," Proc. 5th International Symposium on Advanced Radio Technologies, Boulder, Colorado, USA, Mar. 2003, pp. 105-110

[8] C. Mencar, M. Torselho, D. Dell'Agnello, G. Castellano, C. Castiello, "Modeling user preferences through adaptive fuzzy profiles," Ninth International Conference on Intelligent Systems Design and Applications, 2009. ISDA '09

[9] R. Bell and Y. Koren. "Lessons from the Netflix prize challenge," ACM SIGKDD Explorations Newsletter – Special issue on visual analytics, vol. 9, issue 2, Dec. 2007, pp. 75-79, doi:10.1145/1345448.1345465

[10] A. Toxboe, "Shopping cart design pattern," in UI-Patterns.com. http://ui-patterns.com/patterns/ShoppingCart.