DrIveSCOVER: A Tourism Recommender System Based on External Driving Factors

Benjamin Klotz[1], Pasquale Lisena[1], Raphaël Troncy[1], Daniel Wilms[2], Christian Bonnet[1]

[1]EURECOM, Sophia Antipolis, France, [2]BMW Research, Garching, Germany

Abstract. In this paper, we present the design and implementation of DrIveSCOVER, a recommender system for places and events in case of an in-car use, where the driving conditions such as weather and local traffic are taken into account. We integrate multiple data sources using semantic technologies and we devise recommending functions that are presented in a web-based application. Data is organized according to five root classes: accommodation, car amenity, events, gastronomy and points of interest. An interest score is calculated from the weighted user inputs in terms of preferences of classes and driving conditions. The application is available at `http://drivescover.eurecom.fr/`.

1 Introduction

Recommender systems in the context of driving mobility present a new challenge when taking into account numerous real world external factors. This topic is especially relevant considering the challenge of third-party integration in location-based services that can support the driving experience, as well as semantic mobility learning patterns that use places as nodes between trips [4,6]. This work focuses on the general problem of semantic and location-based recommendations. Another goal is to benefit from the large amount of existing web data describing sights, points of interests and activities. The problem is divided into two sub-questions: *i* How can we integrate external data available in various formats and from different sources? *ii* How can we recommend places and activities for a driver considering the driving context?

We aim to integrate web data [1] and we rely on ontologies describing tourism and travel concepts [3,5] for the selection of various sources. In contrast to existing methods that perform recommendations being content-based, collaborative-filtering or hybrid [2], we define an interest score that takes into account external factors such as the opening hours of businesses, the weather forecasts or the ratings of venues. In the case of a driver using a GPS device, and assuming a route selection, we can easily predict the next location of the car and the recommender system will be able to look for places in a range around the scheduled route with no additional user input.

The remainder of this paper is organized as follows. Section 2 presents our approach for integrating multiple web data sources. Section 3 describes our recommender system and the interest score calculation.

2 Data Selection and Implementation

We selected five root classes that represent the traveler needs:

– Accommodation: integrating data using the Foursquare API[1]
– Gastronomy: integrating data using the Foursquare API
– Car amenity: integrating data using the LinkedGeoData SPARQL endpoint[2]
– Events: integrating data using the OpenAgenda API[3]
– Points of interest: integrating data using the DBpedia SPARQL endpoint[4]

We implemented a single-page web application called DrIveSCOVER[5], built on top of a MEAN stack[6] (MongoDB, ExpressJS, AngularJS, NodeJS). First, the user defines an itinerary from an origin to a destination for a certain date and time. On the server side, the GoogleMaps APIs are used for computing the route that reflects the input. For each route point, we look for nearby places that belong to the selected classes. In order to consider places only around estimated break areas, we look for route points around the areas where the vehicle is expected to be after the regular driving time between breaks $\Delta T \approx 2 hours$ with an uncertainty $\delta\Delta T$. The resulting segments allow us to do query, integrate, score and recommend in a separate way for each cluster. Then, the results are displayed on a map, with the details on each selected place.

If the origin A and the destination B are located in different countries, DrIveSCOVER will take that into account and will retrieve information in English if possible, and in the local language if English is not available.

3 Recommender System

A JavaScript function calculates the recommendation score using different features: matching classes C with weights W_C, traffic quality in the direction of the place t, quality of the weather w based on temperature w_T and humidity w_H, arrival during the opening hours h, and existing rating r.

$$score = [W_C * C] * h * (w + r + t)$$

$$h = \begin{cases} 0 \; if \; arrivalTime \cap opening \; hours = \varnothing \\ 1.5 \; if \; arrivalTime \cap opening \; hours \neq \varnothing \\ 1 \; else \end{cases}$$

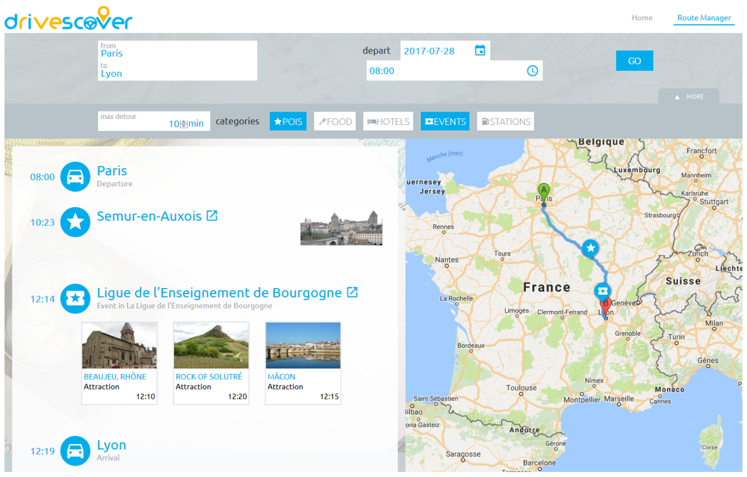
$$r = \begin{cases} 0 \; if \; undefined \\ exisiting \; rating \; between \; 0 \; and \; 1 \; if \; available \end{cases}$$

[1] `https://developer.foursquare.com/`
[2] `http://linkedgeodata.org/About`
[3] `https://openagenda.com/`
[4] `http://wiki.dbpedia.org/`
[5] `http://drivescover.eurecom.fr`
[6] `https://www.mongodb.com/blog/post/the-mean-stack-mongodb-expressjs-angularjs-and`

$$w = \frac{|w_T - w_{Tideal}| + |w_H - w_{THideal}|}{2} \quad w_{Tideal} = 20°C, w_{Hideal} = 50\%$$

Traffic and weather quality are extracted from the Here.com API. Some ratings for Accommodations and Gastronomy places are available for places described in the Foursquare database. In case of Points of Interest, the places are always linked to Wikipedia articles describing them and the page ranking[7] is used as a rating.

If we study a trip from an origin A to a destination B that contains at least one break, and assume that every break i will happen in a short proximity from a town or city C_i. the application will suggest places at the beginning and end of a trip from C_i to C_{i+1}. For instance, a morning trip or an evening trip containing a need for an accommodation will result in different suggestions: in the first configuration, the score will put forward restaurants for a lunch break and avoid cities if there is a break expected during the morning traffic congestion time, while in the second configuration, the same avoidance will happen at the end of the afternoon, and then places of type "accommodation" will likely be suggested.



from Paris
to Lyon
depart 2017-07-28
08:00
max detour 10 min

08:00 Paris
Departure

10:23 Semur-en-Auxois

12:14 Ligue de l'Enseignement de Bourgogne
Event in La Ligue de l'Enseignement de Bourgogne

BEAUJEU, RHÔNE
Attraction
12:10

ROCK OF SOLUTRÉ
Attraction
12:20

MÂCON
Attraction
12:15

12:19 Lyon
Arrival

Fig. 1. Example of a route between Paris and Lyon, looking for Points of Interest and Events

More practically, DrIveSCOVER would recommend two places along a route between Paris and Lyon. It is illustrated in Figure 1 for a journey planned on the 28 July 2017 with sights and events as stop preferences. The first cluster happens in an area with a small density of events and points of interests, so there are no alternatives suggested. In contrast, for the second cluster, numerous choices are proposed in the dense area of Lyon.

[7] `http://people.aifb.kit.edu/ath/`

4 Future Evaluation

In order to validate the accuracy of DrIveSCOVER, users will be asked to fill static forms about places they would like to have been recommended. Since it is a problem of top N recommendations, we calculate the precision and recall defined by: $Precision = \frac{|L_u(N) \cap TS_u|}{N}$ $Recall = \frac{|L_u(N) \cap TS_u|}{|TS_u|}$ where $L_u(N)$ is the recommendation list up to the N^th element, and TS_u is a set of relevant test items. This quantitative evaluation will be completed with a qualitative one with two questions: Were the recommendation surprising? Would you have followed such a recommendation?

We will evaluate DrIveSCOVER on several pre-defined routes:

Origin	Destination	Duration	Number of stops
Paris	Lille	3 hours	1
Paris	Lyon	4.5 hours	2
Nice	Florence	5 hours	2
Paris	Marseille	7 hours	3
Nantes	Strasbourg	8 hours	3

5 Conclusion

We have seen that web data can provide useful information that can be combined to be really beneficial to a driver looking for places and events during breaks on a long trip. In order to improve our data coverage, we plan to use more complex data integration methods that make use of both static and live data streams. In order to increase the accuracy of the recommender system, we plan to use car simulators that generate numerous GPS traces in order to simulate alternative recommendations. Finally, a user evaluation is in progress.

References

1. G. de Melo and K. Hose. Searching the web of data. In *European Conference on Information Retrieval*, pages 869–873, 2013.
2. A. Hinze and S. Junmanee. Travel recommendations in a mobile tourist information system. In *4th International Conference on Innovations in Software Technologies and Automation (ISTA)*, 2005.
3. A. Mathur, K. Akshatha, A. Shastry, and J. Anitha. A survey on existing tourism ontologies. *IJRET: International Journal of Research in Engineering and Technology*, pages 20–23, 2015.
4. C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. Macedo, N. Pelekis, Y. Theodoridis, and Z. Yan. Semantic trajectories modeling and analysis. *ACM Comput. Surv.*, 45(4):42:1–42:32, 2013.
5. K. Prantner, Y. Ding, M. Luger, Z. Yan, and C. Herzog. Tourism ontology and semantic management system : state-of-the-arts analysis. In *IADIS International Conference WWW/Internet*, pages 111–115, 2007.
6. H. Su, K. Zheng, K. Zeng, J. Huang, and X. Zhou. Stmaker: A system to make sense of trajectory data. *Proc. VLDB Endow.*, 7(13):1701–1704, 2014.